Etude de compatibilité de systèmes Kerberos

Projet d'Administration de Réseaux

Philippe Perrin
François Lopitaux

Informatique $3^{ème}$ année
Option Réseaux et Systèmes Répartis
E.N.S.E.I.R.B.

Mars 2002

Table des matières

Table des figures

Introduction

Kerberos Il s'agit d'un protocole d'authentification, développé par le MIT. Il est aujourd'hui présent dans de nombreux réseaux, grâce à son intégration totale dans les domaines Windows 2000. En effet, sa fiabilité et son extensibilité lui ont valu d'être choisi par Microsoft pour devenir le protocole d'authentification des réseaux Windows. On y distingue la notion d'*utilisateurs* (comme les comptes Windows) et de *ressources* (comme les partages de fichiers par exemple). Mais il existe naturellement d'autres implémentations de ce protocole : celle du MIT lui-même, d'autres du monde du logiciel libre (comme Heimdal), ou encore certaines propriétaires (comme SEAM de Sun).

Notre projet Nous nous proposons donc de tester la possibilité, pour ces différentes implémentations, d'interopérer pour permettre à des utilisateurs et des ressources de différentes plateformes (Windows, UNIX) et implémentations (SEAM, MIT, Heimdal...) de s'authentifier les uns auprès des autres. Naturellement, il existe déjà un certain nombre de documents qui traitent ce problème, nous les avons donc examinés pour en dégager un plan de tests significatifs, permettant d'essayer plusieurs types de solutions.

Ce document Il est structuré en deux parties principales. La première concerne le protocole lui-même (présentation générale, déroulement) puis énumère les différentes implémentations que nous avons testées. Cette partie a pour but d'assurer le minimum de connaissances requises pour lire la suite du document.

Celle-ci consiste essentiellement en une énumération commentée des divers tests que nous avons menés. Elle commence par résumer un document de Microsoft qui nous a permis de mieux cibler nos tests d'interopérabilité. Puis, en dédiant un chapitre par type de solution, nous testons les différentes combinaisons d'implémentations pour tester leur comportement. Nous détaillons également la configuration nécessaire à chaque fois, sans toutefois rentrer dans des détails triviaux que la première partie du rapport a déjà expliqués.

Enfin, des annexes viennent compléter ces deux parties. Elles sont régulièrement mentionnées tout au long de ce document, et comportent des informations générales (glossaire, environnement de travail, références de travail, copies d'écran) et des analyses précises de parties de notre travail (logs de connexions, traces réseau, fichier *core*).

Les tests abordés Les tests de la seconde partie sont ordonnés selon une méthodologie introduite par Microsoft. Chacun de ses chapitres se termine par un

tableau récapitulatif des combinaisons d'implémentations testées. Nous constaterons qu'à part quelques exceptions, les solutions proposées se révèlent tout à fait praticables. Nous y faisons régulièrement référence à l'annexe E, car elle contient tous les logs de connexions et fichiers de configuration que nous mentionons dans la deuxième partie de ce document. Le chapitre 7 est dédié à l'explication du mécanisme de *Single Sign On* : c'est un récapitulatif de la démarche à mettre en œuvre, dans des cas concrets (plus que nos maquettes de tests !), pour qu'un utilisateur puisse ne taper son mot de passe qu'une seule fois et malgré tout accéder à des ressources différentes, éventuellement dans des domaines Kerberos différents. Enfin le chapitre 8 examine la question des changements de mots de passe d'une implémentation à l'autre.

Remerciements

Nous remercions tous ceux qui nous ont aidés, de manière souvent très précise, à atteindre le niveau de détails de ce document. Nous avons échangé un grand nombre de *mails* avec la plupart d'entre eux, ainsi que quelques discussions sur *comp.protocols.kerberos* : Wyllys Ingersoll (Sun), Marc Horowitz (MIT), Sam Hartman (MIT), Nicolas Williams (UBS Warburg), Matthew Golczak (Motorola), Tim Mooney (*North Dakota State University*), Donn Cave (*Washington University*), Chris Hallenbeck (AOL), Steve Langasek (Dodds Net), Ann Adams (Ford), Douglas E. Engert (ANL), Andreas Hasenack (Conectiva), Marcus Watts (*Michigan University*).

Première partie

Le procotole Kerberos V5 et ses implémentations

Chapitre 1

Présentation générale

Portée de l'exposé Ce chapitre n'est pas une description complète et détaillée du protocole Kerberos. Il ne s'agit que des principes de base, nécessaires pour comprendre la suite de ce document. Pour une référence exhaustive sur Keberos V5, il faut se reporter à la RFC 1510 : *The Kerberos Network Authentication Service (V5).*

1.1 Tickets

Le protocole Kerberos est un **protocole d'authentification** sur un réseau non sécurisé (où des pirates peuvent capter toutes les données qui circulent), grâce auquel il est également possible de faire partager des clefs de cryptage entre un utilisateur et un service. Tout ce protocole repose sur l'échange de blocs de données nommés *tickets*. Un ticket est un ensemble d'informations, délivrées par une authorité de confiance, pour un client donné, à destination d'un service donné. Le principe général est le suivant :

1. Un client s'authentifie une fois par session auprès d'une autorité de confiance : celle-ci lui délivre un ticket nommé TGT (*Ticket-Granting Ticket*). C'est la seule fois où l'utilisateur aura à s'authentifier durant sa session de travail, tous services confondus.

2. Le client ainsi authentifié souhaite maintenant utiliser un service distant (telnet, ftp, partage de fichiers, imprimante...) pour la première fois de sa session. Il s'adresse alors à l'authorité de confiance en lui présentant son TGT : celle-ci lui délivre un nouveau ticket, dédié au service.

3. Le client s'adresse alors au service, en lui présentant ce nouveau ticket : il est maintenant authentifié auprès de celui-ci. Ce ticket poura être réutilisé jusqu'à sa péremption.

Cette très courte description du protocole permet de comprendre l'utilisation faite des tickets, mais n'explique pas les détails concrets qui font de Kerberos un protocole très sécurisé. Cela fera partie des sections 2.1, 2.2 et 2.3 de cette partie.

1.2 Hypothèses

1.2.1 Les mots de passe

Circulation Une caractéristique de Kerberos est que les mots de passe des utilisateurs *ne circulent pas* sur le réseau, *même cryptés* (sauf bien sûr lors des changements de mots de passe, où des services dédiés permettent aux utilisateurs de les faire circuler cryptés, avec des clefs temporaires, de manière ponctuelle).

La base des mots de passe Les mots de passe sont tous stockés chez une authorité de confiance, nommée le *Key Distribution Center* (KDC). C'est en effet un distributeur de tickets, dans lesquels seront stockées des clefs de cryptage symétrique. Le KDC joue le rôle de deux services :

- Le service d'authentification (AS : *Authentication Service*) : c'est lui qui délivre les TGT (authentification par mot de passe) pour s'adresser au TGS.
- Le service de délivrance de tickets (TGS : *Ticket-Granting Service*) : on s'adresse à lui avec un TGT, en lui demandant un ticket pour un service spécifique (imprimante, partage de fichiers...).

Ces deux services peuvent être localisés sur deux machines distinctes, pour peu qu'elles disposent toutes les deux de la liste des mots de passe. Toutefois, dans nos explications et tests, nous considérons qu'une seule machine joue ces deux rôles (cas fréquent), et l'appellerons "le KDC".

Qui détient un mot de passe ? Chaque utilisateur dispose d'un mot de passe confidentiel, dont nous verrons l'utilisation à la section 2.1. Mais il existe d'autres mots de passe que ceux des utilisateurs : tous les services auprès desquels on souhaite s'authentifier ont également leur propre mot de passe (cela inclut les imprimantes, les serveurs telnet, ftp..., et le service Kerberos lui-même). La base de données des mots de passes mentionnée précédemment contient tous ceux listés ici (utilisateurs et services).

L'ensemble de ces entités qui détiennent un mot de passe s'appellent des *principals* (des "mandants" en français : ce sont les deux partis d'un échange), et possèdent un nom Kerberos normalisé [1].

Changer un mot de passe Nous verrons en 8.1 page 57 comment un utilisateur change son mot de passe. Il s'agit de l'introduction au chapitre 8 de la seconde partie, dédié à l'étude de compatibilité des implémentations concernant cette opération.

Crypter avec un mot de passe Tout au long de ce document, nous considérons qu'il est possible de crypter des blocs de données (comme les tickets, ou des communications réseau) *à l'aide des mots de passe*. Or une opération de cryptage est une procédure mathématique, utilisant des clefs numériques, alors qu'un mot de passe est une chaîne de caractères. Il font donc comprendre que "crypter avec un mot de passe" consiste à crypter des données avec la valeur numérique obtenue à partir d'un mot de passe par une fonction de hachage.

1. voir en section 1.4 pour ces normes

1.2.2 Synchronisation

Nous avons signalé que des tickets circulent sur le réseau, sans qu'aucun mot de passe ne soit échangé. Pour le cas où un pirate obtiendrait une copie d'un ticket, et commence une attaque de décryptage dessus (comme nous le verrons, les tickets sont cryptés), il faut que le ticket soit rendu inutilisable avant que son attaque ne réussisse. Le protocole Kerberos utilise donc une notion de péremption des tickets : un ticket émis sera rendu inutilisable au bout d'une durée fixée par le KDC. Cela nécessite donc que les machines (celles des utilisateurs et celles des services) soient synchronisées entre elles.

Une fois un ticket expiré, les *principals* rejoueront le protocole automatiquement pour obtenir de nouveaux tickets, cryptés avec de nouvelles clefs.

1.3 Clefs

Kerberos est un procotole servant à authentifier des utilisateurs auprès de services (et réciproquement) ; il permet également, une fois la transaction terminée, que les deux partis se trouvent en présence d'une clef de cryptage commune. Elle pourra optionnellement être utilisée pour sécuriser les données échangées. Comme cette clef fait partie du ticket présenté par l'utilisateur, elle expirera avec lui, et sera donc renouvelée par une nouvelle : ce renouvellement régulier des clefs est une autre sécurité pour les données échangées.

1.4 Convention de nommage des *principals*

Comme nous l'avons vu, un *principal* est un nom qui représente un utilisateur ou un service Kerberos. Ces chaînes de caractères respectent certaines conventions. Par exemple, celle associée à l'utilisateur *user* du domaine Kerberos *KB* est simplement *user@KB*.

Lorsqu'un utilisateur souhaite acquérir un ticket pour un service donné, il s'adresse au KDC en spécifiant le *principal* associé au service :

- Le TGT est un ticket de service comme un autre : il correspond au service *krbtgt*. De plus, un TGT est prévu pour être présenté au KDC d'un comaine *KB1* donné (pour acquérir d'autres tickets), mais peut être délivré par le KDC d'un autre domaine *KB2* s'il y a une relation de confiance. Dans ce cas, le *principal* demandé par l'utilisateur est *krbtgt/KB1@KB2*. Souvent, un utilisateur demande un ticket pour son propre domaine, on a donc $KB1 = KB2$.

- Un service qui permet d'obtenir un *shell* sur un serveur *machine* dans un domaine *KB* nécessite un ticket *host/machine@KB*. Celà inclut les services telnet, rlogin, dtlogin...

- Le service FTP nécessite un autre ticket, de la forme *ftp/machine@KB*.

Ces trois types de tickets (*krbtgt/X@X*, host/X@X et ftp/X@X) sont ceux que nous manipulerons le plus souvent au long de ce document. Il est courant, dans les commentaires, d'omettre la partie *@X* lorsque le domaine Kerberos est implicite.

Chapitre 2

Déroulement du protocole

Nous décrivons ici les étapes de l'authentification auprès du KDC puis d'un service. Pour rendre tout cela plus concret, nous proposons en annexe F page 89 des traces UDP et TCP illustrant les échanges qui ont lieu sur le réseau.

2.1 Acquisition d'un TGT

Les différentes sections de ce chapitre sont à lire en suivant conjointement le schéma de la page 12.

2.1.1 Construction du TGT

Supposons qu'un utilisateur souhaite s'authentifier sur son domaine Kerberos. Il dispose pour cela d'un nom d'utilisateur (*username* sur le schéma) et d'un mot de passe. Son client d'authentification (*kinit* par exemple) effectue alors une requête de TGT auprès de l'*Authentication Service* (AS) en lui indiquant le nom d'utilisateur [1] (pas le mot de passe). L'AS reçoit cette demande, et commence alors la construction d'un TGT. Il crée une clef de cryptage symétrique aléatoire, et compose un ticket contenant :

- la date d'émission du ticket
- la durée de vie du ticket
- la clef de cryptage créée.

Ce service d'authentification dispose de sa propre clef de cryptage (c'est un service "comme un autre", avec son propre mot de passe) : il s'en sert pour crypter le ticket, qu'on appelle désormais le TGT (*Ticket-Granting Ticket*). Seul ce service Kerberos sera donc en mesure de lire les informations qui s'y trouvent.

Par ailleurs, le KDC dispose de la liste des mots de passe des utilisateurs, donc aussi de celui qui tente de s'authentifier. Il en extrait alors ce mot de passe, et s'en sert pour crypter la clef de session qu'il a incluse dans le TGT.

1. il est également possible d'effectuer une *pré-authentification* : voir au glossaire

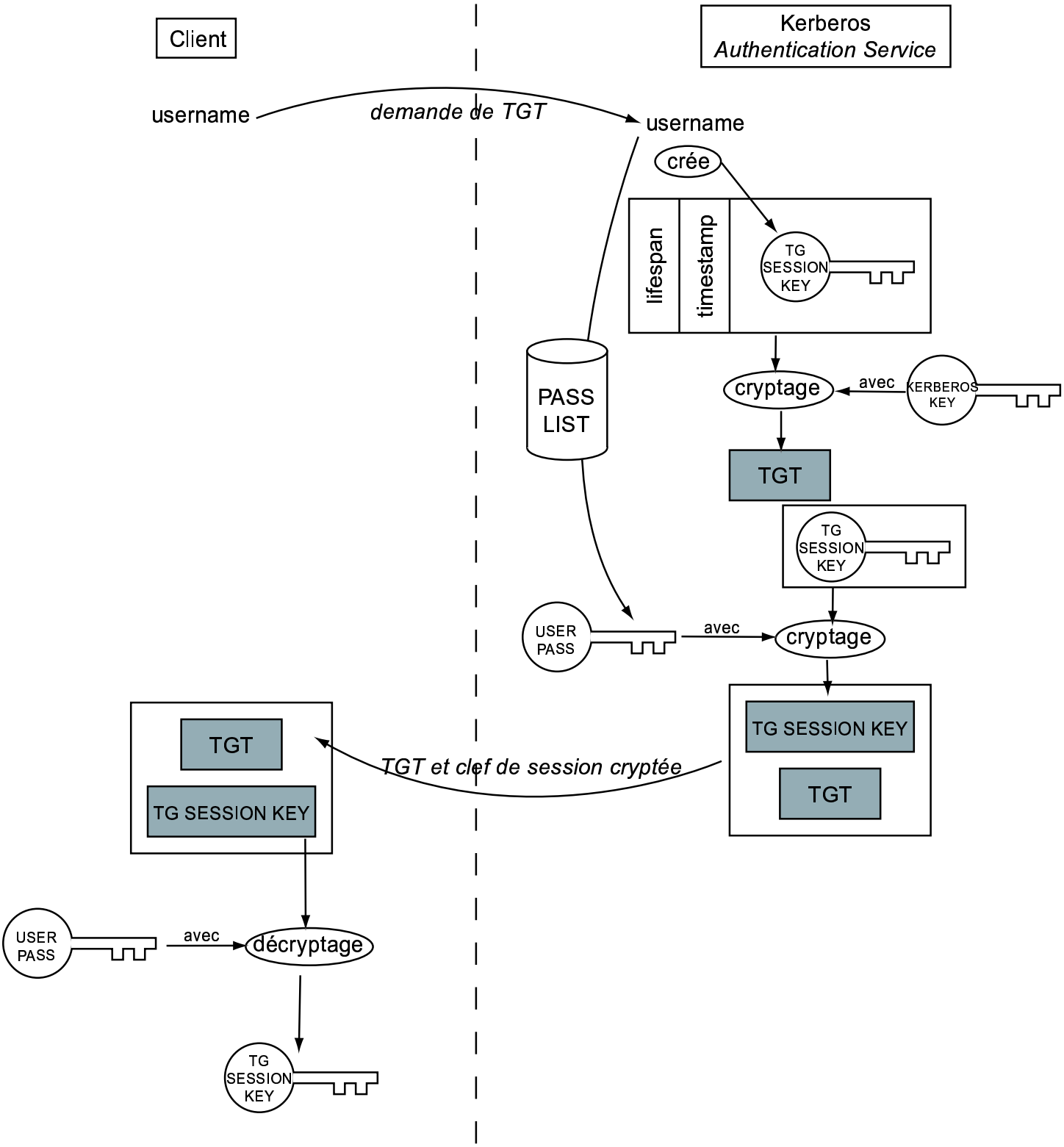

FIG. 2.1 - *Principe de l'acquisition d'un TGT*

2.1.2 Réception du TGT

Le KDC renvoie alors le TGT (crypté avec son propre mot de passe) et la clef de session (cryptée avec le mot de passe de l'utilisateur) au client, à travers le réseau. L'utilisateur, qui a entré son mot de passe sur sa machine locale, peut alors décrypter la clef de session. Il se trouve maintenant en présence :

- d'une clef de session : elle lui servira pour communiquer de manière sécurisée avec le KDC (pour les futures demandes de tickets au TGS) ;
- d'un TGT : il prouve qu'il s'est bien adressé au KDC et a obtenu une clef de session (l'utilisateur présentera ce TGT au TGS).

2.1.3 Attaques

On le voit, un pirate qui obtiendrait une copie des informations diffusées sur le réseau disposerait du même TGT que l'utilisateur, mais ne saurait pas décrypter la clef de session, puisqu'il ne dispose pas du mot de passe de l'utilisateur. Ainsi, le TGT peut être détenu par n'importe qui, mais la clef de session ne sera lisible que par l'utilisateur qui a saisi le vrai mot de passe.

Signalons que le TGT, généré par le KDC, n'est même pas lisible par l'utilisateur, puisqu'il a été crypté avec une clef connue du service Kerberos seulement.

Si un pirate avait voulu se faire passer pour le KDC, il aurait été dans l'impossibilité de lui transmettre une clef de session valide, puisqu'il ne dispose pas du mot de passe de l'utilisateur, utilisé pour la crypter.

2.2 Acquisition d'un ticket de service

Les différentes sections de ce chapitre sont à lire en suivant conjointement le schéma de la page 14.

2.2.1 Construction de l'authentificateur

A présent que l'utilisateur s'est adressé au service AS du KDC (et a donc obtenu un TGT et une clef de session), il souhaite utiliser un des services *kerbérisés* (par exemple, un service telnet, c'est à dire un telnet où il n'aura pas à saisir de mot de passe). Le client va alors composer un authentificateur (*authenticator*). En effet, le TGT ayant pu être intercepté par n'importe qui (mais pas la clef de session), il faut pouvoir prouver son identité.

Cet authentificateur est composé du nom de l'utilisateur, de l'adresse de sa machine, et de l'heure courante. Il crypte ceci avec la clef de session (qu'il est le seul à connaître), et joint ceci au TGT. Il rajoute enfin, en clair, son nom d'utilisateur, son adresse, et bien sûr le nom du service auquel il souhaite accéder.

2.2.2 Authentification du client

Le service TGS du KDC reçoit ces informations, et commence par décrypter le TGT. En effet, il avait été crypté par sa clef personnelle, et si le décryptage réussit, cela prouve son authenticité. Dans ce TGT figure la clef de session avec laquelle le client a crypté l'authentificateur : il peut alors le décrypter. Si

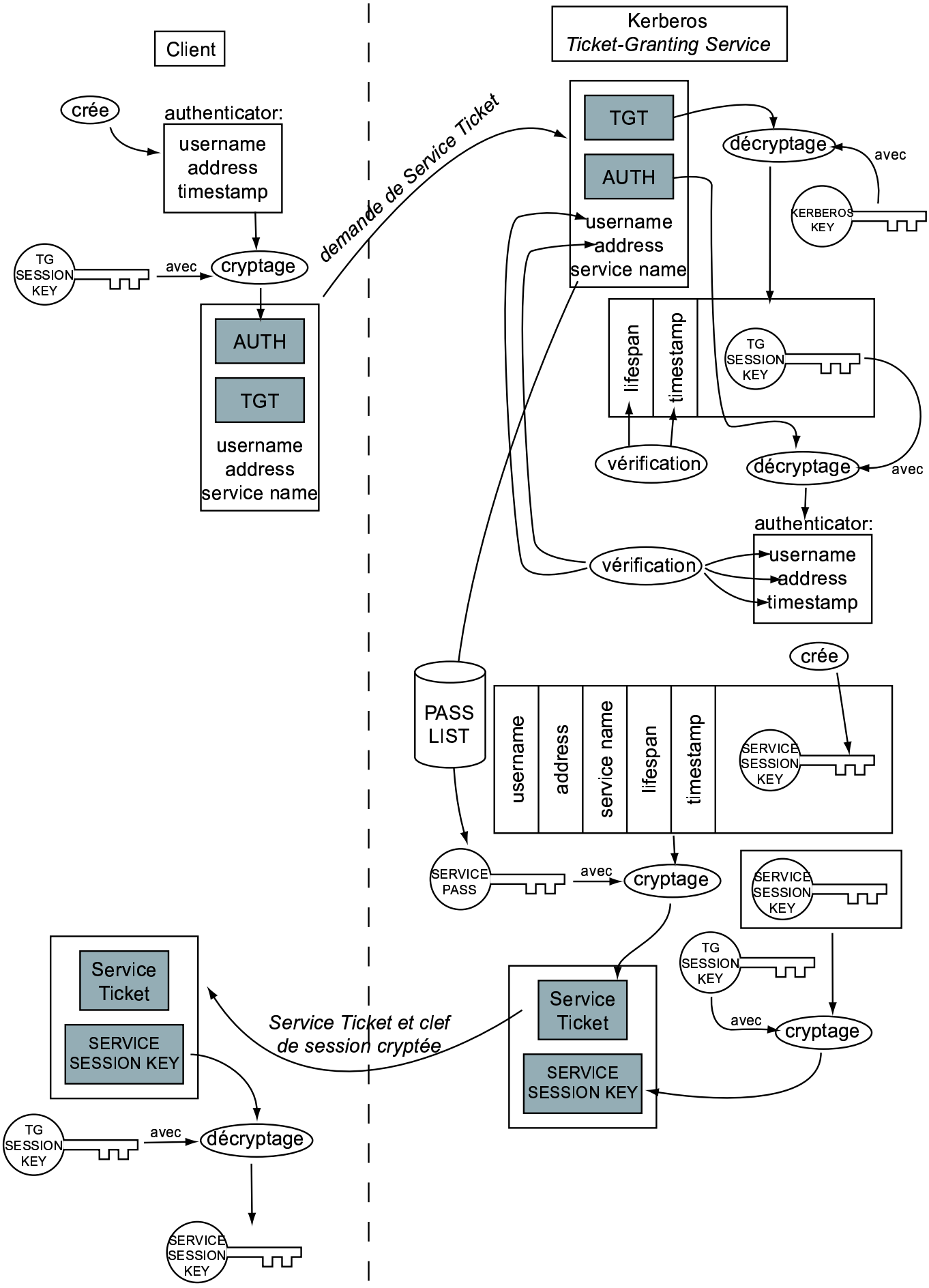

FIG. 2.2 - *Principe de l'acquisition d'un ticket de service*

ce décryptage réussit (c'est à dire que les informations cryptées correspondent aux informations en clair), c'est que le client qui s'adresse à lui est bien celui qu'il prétend être. Par ailleurs, le KDC effectue des vérifications sur les dates et durées de vie, pour s'assurer qu'il ne s'agit pas de tickets ou authentificateurs périmés.

2.2.3 Création du ticket de service

Une fois le client considéré comme authentifié, le KDC crée (aléatoirement) une nouvelle clef de session. Celle-ci servira pour crypter les communications entre le client et le service auquel il souhaite accéder. Il faut donc que cette clef parvienne (de manière sécurisée) au client *et* au service. Voici donc la composition du ticket :

- le nom et l'adresse de l'utilisateur pour lequel le ticket a été créé
- le nom du service que l'utilisateur a demandé d'utiliser
- la date d'émission et la durée de vie du ticket
- la clef de session nouvellement créée pour les communications entre le client et le service.

Pour que le service ciblé soit le seul à pouvoir accéder à ces informations (notamment la clef de session), ce ticket est crypté avec son mot de passe (rappelons que les mots de passe des services sont également stockés dans la base de données du KDC). Il suffira donc que le client donne ce ticket crypté au service pour qu'il puisse en extraire la clef de session.

Enfin, pour que le client puisse également connaître la clef de session (qu'il ne peut bien sûr pas extraire du ticket, ne connaissant pas le mot de passe du service), celle-ci est cryptée avec la clef de session utilisée entre le KDC et le client (qui figurait dans le TGT).

2.2.4 Réception du ticket de service

Le KDC transmet ce ticket de service et cette clef de session cryptée à l'utilisateur. Comme celui-ci connaît la clef de session utilisée entre lui-même et le KDC, il peut décrypter la clef de session qu'il utilisera avec le service ciblé.

2.2.5 Attaques

Un pirate qui aurait "écouté" les échanges entre le client et le KDC aurait pu intercepter le TGT et l'authentificateur. Or nous avons vu en 2.1.3 que ce TGT pouvait déjà être capté, mais non déchiffré : seul le KDC dispose de la clef de cryptage. Quant à l'authentificateur, il ne peut le décrypter sans la clef de session entre l'utilisateur et le KDC. S'il entreprenait de le décrypter avec une attaque de type *brute-forcing*, les informations qu'il contient ne lui seraient d'aucune utilité (nom, adresse, date). Mais en l'ayant décrypté, il aurait trouvé la clef de session : elle aurait déjà probablement expiré, pendant le temps qu'il termine son attaque.

Si un pirate intercepte l'authentificateur et souhaite s'en servir pour s'adresser au KDC (donc rejouer la partie cliente), le KDC détectera l'attaque, car il

retient les *timestamps* utilisés, et donc les réutilisations interdites de ces authentificateurs.

Ainsi, dans le pire des cas, un pirate n'obtiendra qu'une copie du ticket de service, et une copie de la clef de session cryptée (sans pouvoir la décrypter).

2.3 Authentification auprès d'un service

Les différentes sections de ce chapitre sont à lire en suivant conjointement le schéma de la page 17.

2.3.1 Construction de l'authentificateur

Cette section (et la suivante) sont semblables à celles du chapitre précédent. En effet, le problème est le même : il s'agit pour un client de s'authentifier auprès d'un service (qui était le TGS précédemment) à l'aide d'un ticket délivré par le KDC.

A nouveau, le client crée un authentificateur, comportant son nom, son adresse et sa date d'émission. Il crypte cela avec la clef de session générée précédemment par le KDC, spécialement pour les communications entre ce client et ce service. Il envoie alors l'authentificateur crypté et le ticket au service ciblé (un serveur telnet, dans notre exemple précédent).

2.3.2 Authentification du client

Celui-ci réceptionne le ticket de service, qu'il est le seul à pouvoir décrypter. En effet, il connaît son propre mot de passe, avec lequel le KDC avait crypé le ticket. Il en extrait alors les informations qui s'y trouvent, dont la clef de session à utiliser avec le client qui vient de se connecter. Une première vérification consiste à s'assurer que le nom de service figurant dans ce ticket soit bien celui du service adressé. Si c'est le cas, c'est que le ticket émane bien du KDC (il était le seul à pouvoir crypter ce nom avec le mot de passe du service). Il effectue alors les mêmes vérifications de concordance entre la date, l'authentificateur et le ticket, que celles mentionnées en 2.2.2. Le service a alors l'assurance que le client qui s'est connecté est bien celui auquel le ticket a été délivré par le KDC.

2.3.3 Authentification du service

Optionnellement, le service peut s'authentifier auprès du client. Comme il est le seul à disposer de la clef de session, il va crypter une information connue du client, pour que celui-ci vérifie sa valeur. Par exemple, le service va relire le *timestamp* de l'authentificateur (qui est un entier), et crypter cette valeur incrémentée de 1 avec la clef de session.

L'utilisateur réceptionne ce paquet crypté, le décrypte, et doit y lire le même entier (il a mémorisé le *timestamp* inséré dans l'authentificateur). Si c'est bien le cas, le client considère qu'il s'adresse effectivement au service demandé, et il dispose maintenant d'une clef de cryptage, en commun avec lui.

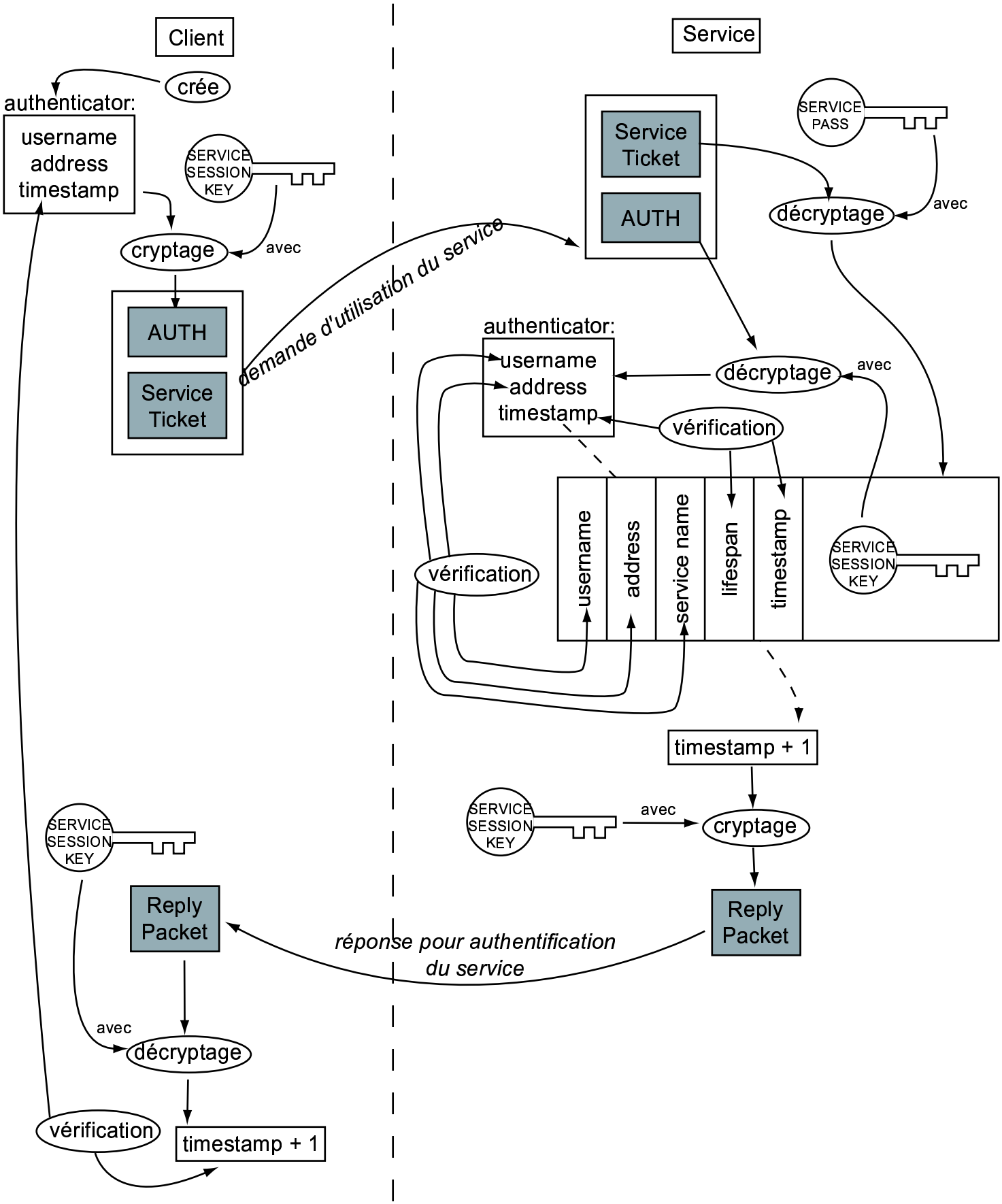

FIG. 2.3 - *Principe de l'authentification auprès d'un service*

2.3.4 Attaques

Un pirate qui voudrait se faire passer pour un utilisateur en s'adressant au service devra lui fournir un ticket contenant le nom de l'utilisateur usurpé. Ne pouvant le générer lui-même (puisqu'il ne peut pas le crypter avec le mot de passe du service), il devra donc en obtenir une copie lors de son passage sur le réseau. Puis il devra fournir un authentificateur. Comme ils ne peuvent être utilisés deux fois, il devra en construire un lui-même. Mais n'ayant pas la clef de session, c'est impossible.

Si un pirate souhaite se faire passer pour le service (en lui volant son adresse, par exemple), il lui sera impossible de décrypter l'authentificateur, et donc de retourner le paquet crypté qui contient $timestamp+1$: le client détectera l'usurpation.

Chapitre 3

Implémentations

Ce protocole d'authentification est (censé être) indépendant de la plateforme et des services. Notre projet est donc de tester l'interopérabilité des implémentations existantes. Certaines sont commerciales, d'autres proviennent du monde libre ; certaines utilisent Kerberos directement dans l'authentification du système, d'autres ne le proposent qu'en option. Nous relatons dans la prochaine partie l'ensemble des tests que nous avons menés entre ces configurations.

3.1 Windows

3.1.1 Domaines Windows 2000

Les domaines Windows 2000 utilisent le protocole Kerberos V5 pour l'authentification des utilisateurs auprès des ressources partagées. Ainsi, un contrôleur de domaine joue le rôle du KDC (AS et TGS) de manière standard.

L'utilisation de Kerberos par les services et clients Windows est complètement transparente pour l'utilisateur : il n'y a donc aucune démarche à suivre pour utiliser le protocole.

Pour une utilisation plus avancée, il existe des outils de Microsoft pour visualiser les tickets détenus par un utilisateur, et administrer directement le KDC.

- en ligne de commande : *klist* (voir écran D.1 page 73)
  `www.microsoft.com/windows2000/techinfo/reskit/tools/existing/klist-o.asp`
- graphique : *kerbtray* (voir écran D.2 page 73)
  `www.microsoft.com/windows2000/techinfo/reskit/tools/existing/kerbtray-o.asp`
- en ligne de commande : en installant le kit *Support Tools* du CD de Windows 2000 (avec `\SUPPORT\TOOLS\SETUP.EXE`), on a accès aux outils *ktpass* et *ksetup*, que nous verrons plus loin dans l'exposé.

Nous supposons à présent que ces derniers outils (*ktpass* et *ksetup*) sont installés sur le KDC Windows 2000.

Ce KDC sous Windows est capable d'utiliser les algorithmes de chiffrement de types DES-CBC-MD5 et DES-CBC-CRC. Il faudra en tenir compte lors de la configuration des machines UNIX (services et clients).

Enfin, nous conseillons d'activer l'audit des *connexions de comptes* sur le contrôleur principal de domaine. De cette manière, il sera facile d'observer toutes les délivrances de tickets aux clients (réussites et échecs).

3.1.2 KTelnet

Windows ne proposant pas de clients telnet et FTP kerbérisés (ni de serveurs, d'ailleurs), nous avons cherché une implémentation cliente de Kerberos V5 compatible avec des services UNIX. Nous utilisons un logiciel nommé *KTelnet*, un outil conçu pour établir des connections telnet et FTP kerbérisées. Il n'utilise pas les éventuels tickets détenus par un utilisateur loggué sur un domaine Windows 2000 ; ce client est donc utilisable sur toute plateforme Win32.

Il est téléchargeable à `www.stacken.kth.se/~thn/ktelnet`. Attention : lors de nos essais, la seule version supportant Kerberos 5 est la version V3.00.950 BETA 010626 (les versions non beta n'utilisent que Kerberos 4). Une fois installé, nous le paramétrons pour correspondre à notre domaine de test. Il n'utilise pas de fichier *krb5.conf* (que nous verrons en 3.2), mais un équivalent sous forme graphique, arborescente, accessible à travers les *Properties*. L'image D.5 page 76 montre un exemple de session telnet kerbérisée (avec quelques octets probablement mal interprétés, si on considère la première ligne du *shell*...).

Concernant le FTP, ce logiciel propose deux interfaces : l'une textuelle (où on entre les commandes FTP comme un client UNIX), l'autre graphique (comme la plupart des clients Windows). Dans ce document, nous ne fournirons que les sorties textuelles de la première interface.

3.2 UNIX

Dans le monde UNIX, les procédures sont beaucoup moins transparentes pour les utilisateurs. La norme du MIT (qui fait référence) définit un ensemble de commandes et de fichiers de configuration, quelle que soit l'implémentation. Les chemins mentionnés ici sont donnés à titre indicatif : ils peuvent varier suivant les distributions.

/etc/krb5.conf C'est le fichier (ASCII) qui mermet de définir, pour un système donné, les différents domaines Kerberos disponibles. Pour chacun d'eux, on spécifie le KDC, le domaine, etc... Il y est également précisé les comportements par défaut des clients et serveurs qui s'exécutent sur le système (domaine par défaut, modes de cryptage, ...). Des exemples de ce fichier seront fournis dans les sections qui nécessitent son paramétrage.

/etc/krb5.keytab Ce fichier (binaire) contient les clefs de cryptage des services kerbérisés. C'est là qu'un tel service (par exemple un serveur telnet) récupère sa clef de décryptage pour pouvoir lire les tickets que lui présenteront les clients. Ce fichier est généré par le KDC, qui a créé la clef, et doit être transmis de manière sécurisée au serveur kerbérisé.

kinit C'est la commande qui permet d'obtenir un TGT auprès du KDC. L'utilisateur fournit son nom et son mot de passe, et il reçoit le ticket. Puis il ne lui sera plus nécessaire de les saisir lorsqu'il s'adressera à un des services kerbérisés du domaine.

klist C'est la commande qui permet de lister les tickets en possession de l'utilisateur. Cela comprend le TGT comme les tickets de service, et le fait qu'ils soient périmés ou non.

kdestroy Cette commande sert à vider le cache qui contient tous les tickets acquis, y compris les TGT.

kadmin Utilisée par un administrateur, cette commande sert à administrer les KDC à distance. *kadmin.local* offre les mêmes services, mais s'exécute en local sur le KDC.

ktutil Cette commande est utilisée pour manipuler les clefs des services kerbérisés. Elle permet notamment d'ajouter des clefs du KDC dans le fichier *krb5.keytab*.

3.2.1 Heimdal

C'est une implémentation libre du protocole Kerberos V5. Disponible sur `www.pdc.kth.se/heimdal`, les fichiers binaires proposés sont compilés pour des distributions Linux et FreeBSD.

Pour nos tests, nous avons téléchargé les packages Debian pour la version 0.2l (la dernière stable disponible lors de la rédaction de ce document). Elle s'installe simplement, sous Debian, avec la commande *apt-get* :

- *apt-get install heimdal-kdc* pour installer le KDC (non utilisé dans nos tests) ;
- *apt-get install heimdal-docs* pour installer la documentation Heimdal (*man* et *info*) ;
- *apt-get install heimdal-servers* pour installer les services kerbérisés de Heimdal (telnet, ftp. . .) ;
- *apt-get install heimdal-clients* pour installer les clients utilisables avec les serveurs kerbérisés.

La plupart de ces packages installeront des dépendances, notamment *heimdal-lib*. L'installation des clients Heimdal supprimera les clients telnet et ftp déjà présents sur le système. En effet, les nouveaux installés supportent aussi bien les serveurs kerbérisés que les serveurs standards.

3.2.2 SEAM

Installation SUN propose sa propre implémentation de Kerberos : *Sun Enterprise Authentication Mechanism* (SEAM), qui fait partie du *Solaris 8 Admin Pack*. Il est téléchargeable à l'adresse `www.sun.com/bigadmin/content/adminPack/` (télécharger et exécuter le fichier *Sun_Enterprise_Authentication_Mechanism.shar*) ; il faut également installer le *Solaris Encryption Pack* pour bénéficier du cryptage Kerberos, téléchargeable à `www.sun.com/solaris/encryption`. La copie d'écran D.4 page 75 montre un exemple de sa configuration ; celle-ci sert à générer automatiquement le fichier *krb5.conf*.

Les clients Les clients kerbérisés de SEAM (telnet, FTP...) portent le même nom que les clients standards, mais sont installés ailleurs (*/usr/krb5/bin*). Pour pouvoir les distinguer facilement (et pour respecter les mêmes conventions de nomage que sous Heimdal), nous avons créé des liens symboliques dans */usr/bin* permettant d'appeler ces nouveaux clients :

```
root     bin        66676 Mar 16  2000 /usr/bin/ftp
root     other         17 Feb  2 14:43 /usr/bin/kftp -> /usr/krb5/bin/ftp
root     other         20 Jan 31 21:38 /usr/bin/ktelnet -> /usr/krb5/bin/telnet
root     bin        74920 Jan  5  2000 /usr/bin/telnet
```

Ce sont ces clients en *k** que nous utiliserons dans les séquences de test dans la suite de ce document.

3.2.3 MIT Kerberos V5

Comme nous le verrons dans les tests [1], l'implémentation SEAM offerte par Sun est relativement inutilisable en termes d'interopérabilité. Nous avons donc cherché une alternative pour Solaris, et avons choisi l'implémentation du MIT. Disponible à `web.mit.edu/kerberos`, elle s'installe facilement (grâce notamment à des binaires précompilés) et donne de meilleurs résultats [2].

En suivant les instructions des manuels disponibles au MIT [3], il est facile de configurer les services Kerberos pour nos tests. Contrairement à SEAM, cette implémentation utilise des fichiers de configuration localisés directement dans */etc* (et non */etc/krb5*).

Attention Lors de notre migration de SEAM au Kerberos du MIT, nous avons rencontré un problème qui pourrait apparaître ailleurs. En tentant une connexion telnet (ou peut-être avec d'autres services ?) d'une machine sur elle-même, il est possible que le KDC ne délivre pas de ticket ! En utilisant *tcpdump*, nous nous rendons compte qu'au lieu de demander le *principal host/thot.mds* au KDC, le client demande *host/thot*. Comme le KDC n'interprète pas la chaîne de caractère demandée, il ne sais pas que *thot* est en fait *thot.mds*. L'erreur vient en fait du client qui demande *thot* alors que la ligne de commande était `telnet -x -l kerby thot.mds`. La solution est d'éditer le fichier */etc/hosts* et de rajouter *.mds* à *thot* (et à mettre à jour */etc/hostname.pcn0*).

1. au chapitre 3 de la deuxième partie
2. Lors de notre installation, la dernière version était la 1.2.3. La suivante, 1.2.4, est parue trop tard pour pouvoir être prise en compte dans nos tests.
3. mentionés dans les références de l'annexe C

Deuxième partie

Etude de compatibilité

Chapitre 1

L'avis de Microsoft

Nous ne sommes pas les premiers à avoir envisagé des études de compatibilité des systèmes Kerberos. Comme ce protocole d'authentification est intégré en natif dans les domaines Windows 2000, Microsoft a déjà mené de tels tests. Leurs conclusions figurent dans un (court) document, téléchargeable à `www.microsoft.com/windows2000/techinfo/howitworks/security/kerbint.asp`

Sans rentrer dans les détails, le tableau 1.1 résume les possibilités d'interactions entre les différentes implémantations de Kerberos.

KDC	Service	Client	Solution
Windows 2000	Windows	Windows	Natif
		UNIX	Configuration du client
	UNIX	Windows	Confiance unilatérale Compte de service
		UNIX	Confiance unilatérale Compte de service
UNIX	Windows	Windows	Confiance bilatérale
		UNIX	Confiance bilatérale
	UNIX	Windows	Configuration du client
		UNIX	Natif

FIG. 1.1 - *Solutions d'interopérabilité entre Windows et UNIX*

Natif Cette situation correspond simplement au cas où le KDC, le service et le client ont été conçus pour être utilisés ensemble (par exemple, un utilisateur Windows qui s'authentifie auprès d'un KDC Windows pour accéder à une ressource Windows). Ce cas ne requiert bien sûr aucune configuration particulière et ne rentre pas dans le cadre de notre projet.

Configuration du client Cela correspond à la seule configuration d'un client pour utiliser les ressources d'un domaine Kerberos homogène (KDC et services) de nature différente. Par exemple, il peut s'agir pour un système Linux d'utiliser un contrôleur de domaine Windows 2000 pour l'authentification, et d'accéder à des ressources Windows.

Confiance unilatérale Cette solution consiste à paramétrer un domaine Kerberos pour qu'il accepte les TGT émis par un autre (en qui il a *confiance*). Par exemple (cas typique), des utilisateurs s'authentifient sur un domaine Windows 2000, et utilisent des services UNIX d'un autre domaine qui fait confiance au premier. Ce deuxième domaine est un *domaine de ressources*.

Confiance bilatérale Semblable à la confiance unilatérale, cette configuration permet à des utilisateurs d'un domaine Windows 2000 d'utiliser des ressources UNIX, et aux utilisateurs UNIX d'utiliser les ressources Windows 2000 (où deux domaines Kerberos, Windows et UNIX, sont en confiance).

Compte de service C'est une solution alternative à la confiance unilatérale. Plutôt que de distinguer deux domaines en confiance (un UNIX et un Windows), les ressources UNIX font partie intégrante de l'unique domaine géré par le KDC Windows 2000. Ces services disposent alors de leur propre *principal* dans la base des "utilisateurs" Windows.

Remarque sur les relations de confiance Lorsqu'un domaine Windows 2000 fait confiance à un domaine UNIX kerbérisé, il faut faire correspondre les utilisateurs UNIX à un ou plusieurs utilisateurs du domaine Windows pour gérer leurs authorisations. On établit alors des relations de *mapping* entre des utilisateurs UNIX et un ou plusieurs utilisateurs Windows. Les tickets provenant du domaine UNIX et présentés à Windows donneront lieu aux restrictions de sécurité prévues pour les utilisateurs Windows mappés.

Voici un exemple (figure 1.2) où un utilisateur d'un domaine A souhaite accéder à une ressource d'un domaine B. Ici, c'est le domaine B qui a confiance en A, puisqu'il accepte des TGT émis par le KDC de A. **[1:]** Le client s'adresse à son propre KDC (du domaine A) pour demander un TGT du domaine B. Après authentification, le KDC le lui retourne. **[2:]** Puis ce client présente ce TGT au KDC B *qui a confiance en A* et donc qui l'accepte comme s'il faisait partie du domaine B : il lui donne un ticket utilisable avec la ressource demandée. **[3:]** Le client présente enfin ce dernier ticket à la ressource, qui l'accepte sans même savoir que ce client provient d'un autre domaine.

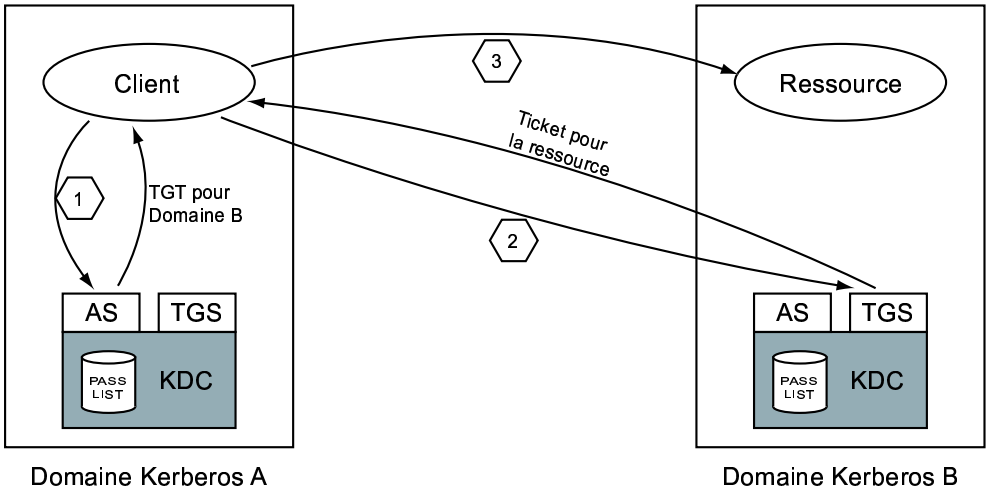

FIG. 1.2 - *Exemple de relation de confiance dans l'accès à une ressource*

Conséquences sur notre étude Ce document de Microsoft nous montrant la marche à suivre, nous en tenons naturellement compte dans le découpage de nos tests de compatiblité. Nous commencerons par des situations où Windows 2000 implémente le KDC :

Chapitre 2 : Nous y traitons des cas où des clients UNIX s'authentifient pour accéder à des ressources du domaine Windows 2000 (solution **configuration de clients**).

Chapitre 3 : Nous y traitons la configurations de services UNIX qui utilisent des *principals* mappés sur des utilisateurs du domaine Windows 2000 (solution **comptes de services**).

Chapitre 4 : Ce chapitre est dédié au recours à une **confiance unilatérale** pour que des ressources d'un domaine UNIX soient accessibles par des clients authentifiés dans un domaine Windows 2000.

Puis nous effectuerons d'autres tests où le KDC est implémenté par un système Solaris :

Chapitre 5 : Combiné avec un domaine de ressources Windows 2000, nous testons la compatibilité des clients variés utilisent la relation de **confiance bilatérale**.

Chapitre 6 : Ici un nous n'utilisons qu'un seul domaine, servant à des ressources UNIX et aux authentifications (solution **configuration de clients**).

Structure des tests Maintenant que la liste de nos essais est établie, précisons leur forme. Dans chacun des chapitres listés ci-dessus, le KDC est fixé, et seules les implémentations des services et clients varient. Pour chacune des combinaisons possibles, nous testons quelques services de base (authentification, telnet, FTP...). Ainsi, la plupart des tests (réussis) se ressemblent : nous ne fournissons pas les sorties d'écran de chacun, mais seulement d'une partie (les plus significatifs), an annexe E. Enfin, chaque série de tests se termine par un tableau récapitulatif avec nos conclusions sur la solution testée.

Quelques noms Enfin, pour se familiariser avec les noms rencontrés dans cette partie, voici une liste des machines et domaines manipulés, et leurs systèmes d'exploitation.

KERBYKB.LOCAL C'est le domaine Windows 2000 que nous utilisons tout au long de nos tests. Son KDC est la machine **kerby.mds**, de nom NetBIOS **KERBY**.

THOTKB C'est le domaine Kerberos implémenté par un KDC sous Solaris 8. Au fil de nos tests, celui-ci utilise d'abord SEAM, puis la version du MIT. Son KDC est **thot.mds**. Il arrivera que ce système fasse temporairement partie du domaine Windows *KERBYKB.LOCAL* lorsque nous en aurons besoin pour nos tests.

amon.mds et verone Ce sont deux PC sous Linux Debian, sur lesquel nous avons installé les services et clients Heimdal. Ils appartiendront aussi bien au domaine *THOTKB* qu'à *KERBYKB.LOCAL*.

Chapitre 2

KDC Windows 2000 et Configuration des Clients

Blocage Au chapitre 1, Microsoft a classé cette solution de configuration des clients (avec un KDC sous Windows 2000) dans le cas où des utilisateurs utilisent des clients UNIX pour accéder à des ressources Windows. Or il nous a été impossible de trouver un couple service – client où le premier serait sous Windows et le second sous UNIX, utilisant l'authentification Kerberos.

Les clients UNIX En effet, les clients kerbérisés sous UNIX sont mal adaptés aux services Windows. Il s'agit essentiellement du telnet, du ftp, de rlogin... tous ces services sont peu ou pas utilisés sous Windows. Il existe bien des services telnet et ftp sous Windows 2000, mais ils ne sont pas kerbérisés (le telnet ne peut supporter que le NTLM, par exemple). La seule possibilité de service à tester est le partage de fichiers de Windows. L'unique version de SAMBA qui soit censée supporter Kerberos V5 est la version 3.0 (la dernière version à ce jour est la alpha 15, disponible par CVS uniquement).

SAMBA Nous téléchargeons donc cette dernière version de SAMBA. Pour être compilée, elle doit disposer de bibliothèques LDAP et Kerberos : nous téléchargeons donc les sources de OpenLDAP 2.0.23 et le Kerberos 1.2.3 du MIT. Nous compilons ces deux modules, puis SAMBA lui-même (spécifier l'option *--with-krb5=<DIR>/krb5-1.2.3/src* au script *configure*).

Problèmes Sur notre système Solaris 8, l'exécution du script *configure* échoue à la fin, lors d'une vérification finale qui consiste à compiler un petit fichier de test. En tentant la compilation "à la main", nous ne constatons aucune erreur, et modifions donc le script pour qu'il continue malgré tout son exécution.

Puis, lors de la compilation de plusieurs exécutables (en fait à leurs éditions de liens), *gcc* affiche un message d'erreur indiquant qu'une fonction (*krb5_free_cksumtypes*) manque au fichier archive *libgssapi_krb5.a*. Celui-ci avait été compilé avec les bibliothèques de Kerberos, et il s'avère qu'il lui manque le module *keyed_checksum_types.o*. Nous le lui ajoutons donc avec la commande :

```
ar q libgssapi_krb5.a crypto/keyed_checksum_types.o
```

Puis nous reprenons la compilation de SAMBA, qui se termine avec succès.

Finalement, nous ne pouvons que constater son incapacité à exploiter les bibliothèques Kerberos. En effet, après nous être authentifiés auprès du KDC (donc en ayant obtenu un TGT), nous tentons d'exécuter *smbclient* avec l'option *-k* (pour utiliser Kerberos V5) : il n'essaie même pas de réclamer un ticket de service auprès du KDC (qui aurait dû être de la forme *kerby$@KERBYKB.LOCAL* pour accéder aux partages de fichiers de l'ordinateur de nom NetBIOS *kerby* dans le domaine *KERBYKB.LOCAL*. Nous avons pourtant suivi les instructions du fichier *docs/textdocs/ADS-HOWTO.txt* de la distribution SAMBA, qui expliquait notamment comment configurer le fichier */usr/local/samba/lib/smb.conf*, où nous avons entré les lignes :

```
realm = KERBYKB.LOCAL
ads server = KERBY
security = ADS
encrypt passwords = yes
```

Pour cela, nous avons bien sûr ajouté l'adresse IP de *KERBY* dans le fichier */etc/hosts* (il s'agit du nom NetBIOS de *kerby.mds*). Il nous a été impossible de contacter qui que ce soit qui ait réussi à utiliser SAMBA avec authentification Kerberos 5 (alors que de manière générale, la plupart de nos problèmes avaient déjà été solutionés par d'autres personnes, joignables dans les *newsgroups*).

Conclusion Cette solution théorique ne trouve pas encore d'application concrète. Le seul service Windows qui aurait pu bénéficier d'un client UNIX se révèle incompatible. En examinant de plus près le document de Microsoft étudié au chapitre 1, dans la section qui traite de ce cas de figure, nous lisons que leur exemple utilise un "*Windows 2000-based file server*" sans préciser quel protocole. Mais il y a également une petite note de bas de page qui lève l'ambiguité : *The Windows Services for UNIX NFS server can be used to implement the file sharing between the Windows file server and the UNIX clients. The current releases of NFS for Windows do not include support for Kerberos V5 authentication. This use of Kerberos authentication in NFS is being standardized in the IETF.*
Donc, lors de la rédaction de ce document (il semblerait au courant de l'année 2000), Microsoft n'a pas testé le partage de fichiers NetBIOS (SAMBA ne proposait pas encore le support Kerberos V5), et ne proposait qu'une hypothétique version de NFS qui n'existait pas encore. Après quelques recherches, il semble qu'elle ne soit toujours pas disponible.

Finalement, il n'est pas surprenant que nous arrivions à cette conclusion : cette configuration (KDC quelconque, ressources Windows, clients UNIX) n'est pas testable par défaut de services kerbérisés.

Chapitre 3

KDC Windows 2000 et Comptes de Services

3.1 Intéractions avec le monde UNIX

Démarche Utiliser la solution des comptes de services de Microsoft [1], avec un KDC Windows 2000, consiste à utiliser des ressources UNIX faisant partie du domaine Windows. Dans les tests qui vont suivre, ce domaine s'appelle *KERBYKB.LOCAL*, et le KDC est localisé sur le PDC (*Primary Domain Controller*) : *kerby.mds* (de nom NetBIOS *KERBY*). Chaque service UNIX kerbérisé doit donc disposer d'un *principal* sur le KDC Windows.

Les comptes de service Nous avons vu dans la section 3.2 page 20 qu'un service telnet sur un serveur *machine* devait disposer d'un *principal* de nom *host/machine*. Or notre KDC est sous Windows, qui n'accepte pas le caractère "/" dans ses noms d'utilisateur. Il faut donc créer un utilisateur dédié à ce service (de nom quelconque), que nous faisons correspondre au *principal* Kerberos *host/machine*. Comme nous le verrons au cas par cas, cette opération s'effectue à l'aide de *ktpass.exe*. De même, un service FTP nécessite un *principal* de nom *ftp/machine*.

Contrôle des authentifications Un moyen d'observer les bonnes authentifications des clients (acquisitions de TGT) consiste à observer le journal d'événements du KDC Windows 2000. Voici un exemple de logs après une exécution réussie de *kinit* sur une machine SEAM, nommée *thot.mds* (dont l'adresse IP est 172.16.8.136), avec l'utilisateur *kerby* :

```
Type de l'événement: Audit des succès
Source de l'événement: Security
Catégorie de l'événement: Connexion de compte
ID de l'événement: 672
Date: 05/02/2002
Heure: 08:30:26
Utilisateur: AUTORITE NT\SYSTEM
```

1. introduite au chapitre 1

```
Ordinateur: KERBY
Description:
Ticket d'authentification accordé:
  Nom de l'utilisateur: kerby
  Nom de domaine Kerberos fourni: KERBYKB.LOCAL
  ID de l'utilisateur: KERBYKB\kerby
  Nom du service: krbtgt
  No du service: KERBYKB\krbtgt
  Options du ticket: 0x40800000
  Type de cryptage du ticket: 0x3
  Type de pré-authentification: 2
  Adresse du client: 172.16.8.136
```

Remarque sur Solaris Chronologiquement, pour tester Solaris, nous avons d'abord installé SEAM : étant proposé par Sun, nous pensions faire un bon choix. Malheureusement, ses résultats (relatés dans ce chapitre) se sont avérés très médiocres (voire nuls) en terme d'interopérabilité avec les autres implémentations.

Nous avons donc opté pour une autre solution sous Solaris : la version provenant du MIT lui-même. Les tests qui suivent indiquent donc toujours nos premiers résultats avec SEAM, puis avec l'implémentation du MIT. Nous n'avons pas essayé de faire interagir ces deux ensemble, puisque la plupart des autres tests nous font conclure que la version SEAM est peu compatible avec les autres, et donc d'un faible intérêt pour notre problème.

3.2 Les services Heimdal sous Linux

Acquisition d'un TGT Un premier test, pour vérifier la compatiblité du KDC Windows 2000 avec un client Heimdal[2], a consisté à obtenir un TGT. Nous avons donc créé un utilisateur *kerby* sur *KERBY*, avec un mot de passe utilisateur du domaine Windows 2000. Puis, sous Linux (ici une machine de test *verone* où les clients Heimdal ont été installés et configurés), nous avons exécuté les commandes suivantes :

```
verone:/home/noroot# kinit kerby
kerby@KERBYKB.LOCAL's Password:
verone:/home/noroot# klist
Credentials cache: FILE:/tmp/krb5cc_0
        Principal: kerby@KERBYKB.LOCAL

  Issued           Expires          Principal
Jan 29 00:32:23  Jan 29 10:32:02  krbtgt/KERBYKB.LOCAL@KERBYKB.LOCAL
```

L'authentification initiale a réussi, et le TGT a bien été délivré à l'utilisateur. Ce premier service est donc capable de fonctionner entre un KDC Windows 2000 et un client Heimdal. Pour configurer Heimdal, notre */etc/krb5.conf* a été copié sur l'exemple 1 de la page 78.

2. voir la description de Heimdal en 3.2.1 page 21

Configuration du telnet A présent, le serveur telnet installé par Heimdal sur le serveur Linux *amon.mds* doit disposer d'un *principal* de nom *host/amon.mds*. Sur le KDC Windows, nous créons l'utilisateur *hostamon* (ce nom peut être quelconque), et nous le mappons avec le *principal* :

```
C:\>ktpass -princ host/amon.mds@KERBYKB.LOCAL -mapuser hostamon
           -pass host -out amon.keytab
```

Le mot de passe *host* indiqué ici est le même que celui que nous avons entré lors de la création du compte *hostamon*. Cette commande permet d'extraire la clef de cryptage associée à l'utilisateur *hostamon* dans un fichier binaire *amon.keytab* (de nom quelconque) ; elle établit aussi le *mapping* d'utilisateur entre le nom Kerberos *host/amon.mds@KERBYKB.LOCAL* et l'utilisateur Windows *hostamon*.

Il faut ensuite transmettre le fichier de manière sécurisée sur *amon.mds* pour qu'il serve de fichier */etc/krb5.keytab*. A ce stade de nos tests, ce fichier n'existe pas encore, il suffit donc de déplacer et renommer celui créé par Windows. Nous verrons plus bas (paragraphe de configuration du FTP) comment concaténer un fichier (généré par Windows) à un fichier *krb5.keytab* déjà existant.

```
amon:~# mv amon.keytab /etc/krb5.keytab
amon:~# ktutil list
Version  Type             Principal
     1   des-cbc-crc      host/amon.mds@KERBYKB.LOCAL
```

La commande *ktutil* nous confirme, en lisant */etc/krb5.keytab*, que la clef générée par le KDC est bien installée sur le système *amon.mds*.

Configuration du FTP Le FTP est un autre exemple d'interaction entre Heimdal et un KDC Windows. Pour pouvoir fonctionner, il nous faut générer un *principal* de nom *ftp/machine*. Nous créons donc sur le KDC Windows un utilisateur *ftpamon* et le mappons avec *ftp/amon.mds* :

```
C:\>ktpass -princ ftp/amon.mds@KERBYKB.LOCAL -mapuser ftpamon
           -pass ftp -out amon.keytab
```

Puis nous transférons ce nouveau fichier *amon.keytab* sur *amon.mds*, et le concaténons au fichier *krb5.keytab* avec la commande *ktutil*.

```
root@amon:~# ktutil list
Version  Type             Principal
     1   des-cbc-crc      host/amon.mds@KERBYKB.LOCAL
root@amon:~# ktutil copy amon.keytab /etc/krb5.keytab
root@amon:~# ktutil list
Version  Type             Principal
     1   des-cbc-crc      host/amon.mds@KERBYKB.LOCAL
     1   des-cbc-crc      ftp/amon.mds@KERBYKB.LOCAL
```

La première et la troisième commandes (`ktutil list`) ne servent qu'à illustrer l'état du fichier */etc/krb5.keytab*. L'intégration du nouveau *principal* est assurée par l'exécution de `ktutil copy`.

3.2.1 Les clients Heimdal sous Linux

Services Heimdal - Clients Heimdal

Le service telnet A présent, un utilisateur qui utilise le client telnet de Heimdal peut utiliser son service. Il faut évidemment qu'il existe un utilisateur *kerby* en local sur la machine cible *amon.mds* (le mapping entre l'utilisateur Kerberos *kerby@KERBYKB.LOCAL* et l'utilisateur Linux *kerby* de *amon.mds* est spécifié en ligne de commande). Après l'exécution de *kinit* vue en 3.2, toujours sur la même machine cliente *verone*, nous pouvons effectuer le test. Les résultats figurent dans l'exemple 1 de la page 80.

On observe bien que suite à l'acquisition du TGT en 3.2, le client authentifié *kerby* n'a eu qu'à exécuter le client telnet, sans avoir à re-taper de mot de passe (il n'a eu qu'à spécifier son nom d'utilisateur avec l'option *-l*). Il dispose alors d'un *shell* au nom de l'utilisateur *kerby*.

Le service FTP Disposant toujours du TGT acquis ci-dessus, nous essayons à présent le service FTP kerbérisé depuis la même machine cliente *verone* sans avoir à saisir de mot de passe dans la session FTP. Les résultats figurent dans l'exemple 1 page 84.

A nouveau, il reste nécessaire de saisir le nom d'utilisateur local à *amon.mds*, pour des questions de mapping. On peut constater que tout se déroule bien : l'authentification, l'envoi de commandes, et la réception de données.

3.2.2 Les clients SEAM sous Solaris

Services Heimdal - Clients SEAM

Nous avons installé les services SEAM[3] sur une machine *thot.mds*. Après modification du fichier */etc/krb5/krb5.conf* généré par l'installation, il contient notamment les informations de l'exemple 2 page 79.

Le service telnet Malheureusement, le client telnet SEAM ne semble pas accepter la configuration Windows – Heimdal – SEAM. En effet, l'exemple 2 de la page 80 montre le résultat de la connexion : le client génère un *segmentation fault* une fois la connexion établie (voir en annexe G page 96 pour l'analyse du fichier *core*).

En essayant la plupart des options du client telnet, nous arrivons à la conclusion que c'est l'authentification Kerberos qui pose problème (alors que le ticket a bien été envoyé par le KDC lors de la commande). En effet, en exécutant *ktelnet* sans avoir de TGT, le serveur *amon.mds* propose une authentification non kerbérisée, ce qui fonctionne bien. De plus, l'option *-K* du client telnet permet de s'authentifier sans immédiatement démarrer un shell : cet essai échoue aussi.

Le service FTP Le telnet se révélant inutilisable, nous essayons d'établir une session FTP, toujours vers notre serveur Heimdal. Le résultat de la commande *kftp* (le TGT étant toujours présent) figure en exemple 2 page 84.

On le voit, tout se passe bien pour un client FTP SEAM s'adressant au service Heimdal (authentification, etc...). Remarquons tout de même la ligne `230 Dumpucko!` : elle n'apparaissait pas dans la session FTP initialisée par le client Heimdal en 3.2.1. Après vérification, il s'avère que le serveur envoie ce "commentaire" lorsque le client lui fournit un mot de passe alors que c'était

3. voir la description de SEAM en 3.2.2 page 21

inutile. Il semblerait que le client agisse systématiquement comme cela pour des raisons de compatibilité avec certains serveurs. Ce mot de passe aléatoire est bien sûr ignoré.

3.2.3 Les clients du MIT sous Solaris

Services Heimdal - Clients MIT

Le service telnet En paramétrant un fichier */etc/krb5.conf* semblable à celui de l'exemple 2 page 79, nous pouvons tester le client du MIT [4]. Cet essai se révèle bien plus concluant ; il figure en exemple 3 page 80 ; l'utilisateur dispose bien d'un *shell* sur *amon.mds*.

En exécutant *klist*, nous constatons que le client a bien demandé le ticket *host/amon.mds@KERBYKB.LOCAL* au KDC.

Le service FTP De même, nous observons que le FTP ne pose aucun problème (authentification, commandes, données) : l'exécution figure en exemple 3 page 85.

Ici aussi, l'interaction fonctionne bien. Observons que le paramètre `-x` active l'encryption des communications avec succès (confirmée par le serveur).

3.2.4 Les clients KTelnet sous Windows

Services Heimdal - Clients KTelnet

Le service telnet Nous avons configuré KTelnet [5] avec les mêmes paramètres que le client Heimdal : cela a consisté à recopier le fichier *krb5.conf* de l'exemple 1 page 78 dans les propriétés du client.

Mis à part quelques octets mal transférés entre ce client et le service Heimdal, les résultats de compatibilité sont concluants. En effet, la copie d'écran D.5 page 76 illustre la réussite de l'authentification ; et nous avons pu utiliser le *shell* obtenu sur le serveur sans problème.

Le service FTP Malgré son nom, KTelnet est également un client FTP. Il présente la même compatibilité avec le service FTP de Heimdal que pour le telnet. En effet, nous listons une session FTP réussie (l'authentification pour l'obtention du TGT s'effectue graphiquement) en exemple 4 page 85. Ce client marche donc aussi bien que le client Heimdal.

3.3 Les services SEAM sous Solaris 8

Configuration du telnet A présent, nous essayons les services kerbérisés de SEAM, toujours avec notre KDC sous Windows 2000. Tout comme nous avions mappé un *principal* de nom *host/amon.mds* pour le telnet de *amon.mds*, nous créons maintenant un utilisateur *hostthot* pour le telnet sur le serveur *thot.mds* ; puis nous le mappons avec le *principal* de nom *host/thot.mds* :

```
C:\>ktpass -princ host/thot.mds@KERBYKB.LOCAL -mapuser hostthot
           -pass host -out thot.keytab
```

4. voir la description du Kerberos du MIT en 3.2.3 page 22
5. voir la description de KTelnet en 3.1.2 page 20

Nous transférons ce fichier *thot.keytab* sur la machine cible, où nous utilisons la commande *ktutil* de SEAM pour l'enregistrer.

```
root@thot:/# ktutil
ktutil:  rkt thot.keytab
ktutil:  list
slot KVNO Principal
---- ---- ---------------------------------------------------------
   1    1              host/thot.mds@KERBYKB.LOCAL
ktutil:  wkt /etc/krb5/krb5.keytab
```

La commande *rkt* lit le fichier transféré et le charge dans la mémoire du processus *ktutil* (ce qui n'a aucune influence sur le serveur telnet). Exécuter *list* permet de s'assurer que le fichier généré par Windows 2000 contient bien le bon *principal*. Enfin nous enregistrons cette clef dans le fichier cible */etc/krb5/krb5.keytab* à l'aide de la commande *wkt*. A présent, le serveur telnet dispose de sa clef secrète.

Configuration du FTP Cela consiste à créer un utilisateur mappé sur un *principal* sur le KDC Windows, et à l'ajouter au *krb5.keytab* de *thot.mds*.

```
C:\>ktpass -princ ftp/thot.mds@KERBYKB.LOCAL -mapuser ftpthot
           -pass ftp -out thot2.keytab
```

Puis, sur *thot.mds* (sous Solaris) :

```
ktutil:  rkt thot2.keytab
ktutil:  list
slot KVNO Principal
---- ---- ---------------------------------------------------------
   1    1               ftp/thot.mds@KERBYKB.LOCAL
ktutil:  wkt /etc/krb5/krb5.keytab
```

La dernière ligne a pour effet d'ajouter au fichier *krb5.keytab* (sans l'écraser) les clefs chargées en mémoire, à savoir le contenu du fichier *thot2.keytab*, généré par Windows.

3.3.1 Les clients SEAM sous Solaris

Services SEAM - Clients SEAM

Le service telnet Aussi surprenant que cela paraisse, nous n'arrivons pas à utiliser le client SEAM avec le serveur telnet SEAM ! Nous pensons donc qu'il s'agit d'une incompatibilité avec le KDC Windows 2000 [6]. Les résultats obtenus à l'écran sont rigoureusement les mêmes que ceux obtenus avec un serveur telnet Heimdal, comme en 3.2.2 (*segmentation fault*, et mêmes tickets acquis), dont la sortie figure en exemple 2 page 80.

Le service FTP Nous essayons ensuite ce service avec le client de SEAM, sous Solaris. Le résultat figure en exemple 5 page 86.

Le résultat est bien celui attendu : la seule authentification par *kinit* au paragraphe précédent a suffit à s'authentifier auprès du service FTP (en spécifiant de nouveau le nom d'utilisateur). Quant à l'établissement d'une connexion de données, il s'est bien passé.

6. rappelons qu'en 3.2.2, la combinaison d'un client SEAM avec un service Heimdal a échoué

3.3.2 Les clients Heimdal sous Linux

Services SEAM - Clients Heimdal

Le service telnet Utiliser le client Heimdal ne résout pas le problème du client SEAM. Les même commandes avec Heimdal sous Linux donnent le même résultat, montré en exemple 4 page 81 (sans toutefois générer de *segmentation fault*) : le serveur rompt la connexion.

Le service FTP Quant au FTP, le résultat est une erreur surprenante. La séquence de commandes figure en exemple 6 page 86.

Ici, l'authentification s'est bien passée (étant donnée la ligne `GSSAPI user kerby@KERBYKB.LOCAL is authorized as kerby`), mais la demande d'une connexion de données (pour le retour du *ls*) échoue (le serveur réclame à nouveau une authentification). De même, de simples commandes (comme *pwd*) retournent le même message. En tentant à nouveau l'authentification (commande *USER*), il n'est pas demandé de mot de passe, et l'erreur reste la même :

```
ftp> user
(username) kerby
S:232 GSSAPI user kerby@KERBYKB.LOCAL is authorized as kerby
ftp> ls
S:200 PORT command successful.
S:530 Please login with USER and PASS.
ftp>
```

3.3.3 Les clients KTelnet sous Windows

Services SEAM - Clients KTelnet

Le service telnet Encore une fois, le serveur SEAM rompt la connexion. Voici les logs obtenus lors des essais :

```
[ TerminalId: <xterm> ]
[ Trying mutual KERBEROS5 ]
[ Sent Kerberos V5 credentials to server ]

[ Connection closed ]
```

A l'issue de cet essai, le TGT et le ticket de service pour *host/thot.mds* ont bien sûr été acquis par le client (ces transmissions de tickets ne dépendent que du KDC), mais on constate à nouveau que le serveur s'est déconnecté lors de la réception du ticket de service.

Le service FTP Ce client FTP donne les mêmes mauvais résultats que celui de Heimdal : les messages loggés sont exactement les mêmes qu'à l'exemple 6 page 86. Le serveur prétend autoriser la connexion, puis refuse d'établir des connexions de données.

3.4 Les service du MIT sous Solaris 8

Configuration du telnet et du FTP Ces services s'installent exactement comme ceux de SEAM en 3.3. La génération des *principals* sous Windows, leur copie et intégration sous Solaris sont rigoureusement identiques.

3.4.1 Les clients du MIT sous Solaris

Services MIT - Clients MIT

Le service telnet L'acquisition du TGT et du ticket *host/thot.mds* se passe bien[7]. Le résultat figure en exemple 5 page 81. L'implémentation du MIT ne reproduit donc pas le *segmentation fault* de SEAM[8].

Le service FTP Nous testons donc maintenant le service FTP (qui fonctionnait correctement avec SEAM sous cette configuration en 3.3.1). Le résultat est aussi concluant (celui avec SEAM figurait en exemple 5 page 86). Migrer de SEAM au Kerberos du MIT ne pose donc aucun problème d'interopérabilité supplémentaire.

3.4.2 Les clients Heimdal sous Linux

Services MIT - Clients Heimdal

Les clients Heimdal semblent bien mieux fonctionner avec le serveur du MIT qu'avec celui de SEAM[9]. La connexion telnet est similaire à l'exemple 5 page 81, le FTP à celui de l'exemple 6 page 86.

Rappelons que les clients Heimdal étaient inutilisables avec les services SEAM dans un domaine Windows 2000. Cette implémentation du MIT corrige donc ce problème.

3.4.3 Les clients KTelnet sous Windows

Services MIT - Clients KTelnet

De même, KTelnet fonctionne correctement (avec cryptage des données) avec ce service du MIT. La sortie produite pour le telnet figure en exemple 6 page 86 : l'authentification se déroule bien, et le cryptage des données est activé.

De plus, le FTP ne pose aucun problème. L'authentification, le cryptage des données et les ouvertures de connexions de données fonctionnent.

3.5 Conclusion

Pour conclure ces essais du KDC Windows 2000 avec des services UNIX, dans la solution des comptes de services, le tableau 3.1 (page 37) récapitule nos résultats.

On le voit, les services SEAM posent un réel problème de compatibilité ! Nous décidons de le remplacer par la solution du MIT ; donc nous disposons d'une solution effective pour chaque combinaison de systèmes d'exploitation. Ce mode de compatibilité par comptes de services est donc satisfaisant pour un KDC sous Windows 2000. Nous choisissons d'abandonner les tests d'interopérabilité de SEAM au profit du MIT.

7. voir en 3.2.3 page 22 pour un petit problème de DNS que nous avons rencontré
8. ce problème a été vu en premier en 3.2.2 page 32
9. la combinaison d'un client Heimdal avec un serveur SEAM figure en 3.3.2 page 35

			Résultats	
KDC	Services	Clients	FTP	telnet
Windows 2000	Heimdal	Heimdal	√	√
		SEAM	√	X
		MIT	√	√
		KTelnet	√	√
	SEAM	Heimdal	X	X
		SEAM	√	X
		KTelnet	X	X
	MIT	Heimdal	√	√
		MIT	√	√
		KTelnet	√	√

FIG. 3.1 - *Conclusions sur les comptes de services Windows 2000*

Chapitre 4

KDC Windows 2000 et Confiance Unilatérale

4.1 Rappel

La confiance unilatérale Rappelons qu'il s'agit d'une solution alternative à celle du chapitre 3. Ici, nous considérons deux domaines Kerberos distincts. L'un sert à l'authentification des *utilisateurs* (KDC implémenté par Windows 2000), l'autre est un domaine de *ressources* (KDC implémenté par Solaris, par exemple). Pour que les utilisateurs du premier puissent utiliser les ressources du second, il faut établir une relation de confiance unilatérale ; c'est ce procédé que nous testons à présent.

Solaris 8 Nous avons choisi d'utiliser Solaris pour implémenter le KDC du domaine de ressources. Au cours des tests du chapitre 3, nous avons conclu aux meilleurs résultats du MIT par rapport à SEAM. Ainsi, dans ce chapitre :

- Les utilisateurs appartiennent à un domaine Windows 2000, qui représente notre premier domaine Kerberos : *KERBYKB.LOCAL* ;
- Les ressources UNIX appartiennent à un second domaine *THOTKB* dont le KDC est implémenté par le Kerberos 5 du MIT sous Solaris. Celui-ci fait donc confiance au premier domaine.

4.2 Etablir la relation de confiance

Nous ne donnons pas ici toutes les étapes de configuration d'un domaine Kerberos du MIT, mais seulement les étapes relatives à l'interopérabilité. Pour une description complète, se repporter au chapitre 4 du manuel d'installation du MIT[1]*.*

Le KDC du MIT Notre système Solaris, *thot.mds*, doit maintenant être configuré comme KDC d'un domaine propre : *THOTKB*. Pour cela, nous com-

1. cité dans les références de l'annexe C

mençons par initialiser la base de données des *principals* du domaine (il faut créer le répertoire adéquat en premier) :

```
root@thot:/# mkdir /usr/local/var/krb5kdc
root@thot:/# kdb5_util create -r THOTKB -s
Initializing database '/usr/local/var/krb5kdc/principal' for realm 'THOTKB',
master key name 'K/M@THOTKB'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:
Re-enter KDC database master key to verify:
```

Nous paramétrons alors *thot.mds* pour utiliser ce nouveau KDC qu'il implémente : nous adaptons son */etc/krb5.conf*, comme à l'exemple 3 page 79.

Les autres machines UNIX impliquées devront également utiliser ce fichier de configuration pour savoir accéder aux services du domaine *THOTKB*. Celles où l'authentification s'effectuera sur le domaine Windows *KERBYKB.LOCAL* peuvent garder `default_realm = KERBYKB.LOCAL`, ou alors devront exécuter explicitement *kinit user@KERBYKB.LOCAL*. De plus, grâce à la section `[domain_realm]`, un client qui souhaite accéder, par exemple, au telnet de *thot.mds*, saura qu'il doit s'adresser au KDC de *THOTKB* pour obtenir le ticket *host/thot.mds@THOTKB*.

La relation de confiance Il faut maintenant que le domaine Kerberos ainsi créé fasse confiance au domaine Windows *KERBYKB.LOCAL*. Nous commençons par définir le KDC du domaine *THOTKB* aux stations Windows (cette configuration nécessite un redémarrage de Windows) :

```
C:\>ksetup /addkdc THOTKB thot.mds
```

Puis il faut que le KDC Windows (donc le contrôleur de domaine Windows 2000) tienne compte de la relation de confiance. Nous y exécutons alors la commande *ksetup* donnée ci-dessus, et le redémarrons. Puis nous ouvrons la console MMC *Domaines et approbations Active Directory* : nous sélectionnons les propriétés du domaine *KERBYKB.LOCAL* puis l'onglet *Approbations* (voir une copie d'écran page 77). Nous ajoutons le domaine *THOTKB* à la liste *Domaines qui approuvent ce domaine*, en y entrant un mot de passes secret. Windows signale alors qu'il s'agit pas d'un domaine Windows, nous ignorons bien sûr le message en cliquant sur *OK*.

Puis, il faut que le KDC du MIT connaisse aussi ce mot de passe. Nous y exécutons alors la commande suivante :

```
root@thot:/# kadmin.local
kadmin.local:  ank krbtgt/THOTKB@KERBYKB.LOCAL
WARNING: no policy specified for krbtgt/THOTKB@KERBYKB.LOCAL; defaulting to no policy
Enter password for principal "krbtgt/THOTKB@KERBYKB.LOCAL":
Re-enter password for principal "krbtgt/THOTKB@KERBYKB.LOCAL":
Principal "krbtgt/THOTKB@KERBYKB.LOCAL" created.
```

Ceci crée le *principal krbtgt/THOTKB@KERBYKB.LOCAL* qui représente les TGT pour le domaine *THOTKB* mais délivrés par le KDC du domaine

KERBYKB.LOCAL. Le mot de passe tapé ici est le même qu'entré précédemment sous Windows. C'est la connaissance commune de ce mot de passe entre les deux KDC qui symbolise la relation de confiance.

Confiance Unilatérale – Bilatérale Nous avons déduit cette procédure de confiance unilatérale en lisant plusieurs documents sur Internet. Or aucun ne mentionne la confiance *unilatérale* : tous expliquent comment établir une relation *bilatérale*, même les pages de Microsoft qui préconise pourtant une confiance unilatérale pour cette solution. Nous analyserons l'autre forme de confiance au chapitre 5.

Afin de respecter le *principe de moindre privilège* (n'accorder que les droits strictement nécessaires), nous n'avons donc effectué ici que la moitié des démarches relatées dans les documents, empêchant donc des utilisateur du domaine *THOTKB* de se servir des ressources de *KERBYKB.LOCAL*.

4.3 Les services du MIT sous Solaris 8

Configuration des services Nous créons à présent les *principals* sur le KDC des ressources (celui du domaine *THOTKB*). Comme notre seul système Solaris implémente à la fois le KDC et les services applicatifs (telnet, FTP), ces *principals* sont de la forme *.../thot.mds@THOTKB*.

```
root@thot:/# kadmin.local
kadmin.local:  ank -randkey host/thot.mds@THOTKB
WARNING: no policy specified for host/thot.mds@THOTKB; defaulting to no policy
Principal "host/thot.mds@THOTKB" created.
kadmin.local:  ank -randkey ftp/thot.mds@THOTKB
WARNING: no policy specified for ftp/thot.mds@THOTKB; defaulting to no policy
Principal "ftp/thot.mds@THOTKB" created.
```

Puis il faut les inscrire dans le fichier */etc/krb5.keytab* du serveur *thot.mds*. Comme il s'agit à nouveau du KDC *thot.mds*, les commandes sont simplement (à la suite des commandes précédentes) :

```
kadmin.local:  ktadd host/thot.mds@THOTKB
Entry for principal host/thot.mds@THOTKB with kvno 3, encryption type Triple DES cbc
      mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/thot.mds@THOTKB with kvno 3, encryption type DES cbc
      mode with CRC-32 added to keytab WRFILE:/etc/krb5.keytab.
kadmin.local:  ktadd ftp/thot.mds@THOTKB
Entry for principal ftp/thot.mds@THOTKB with kvno 3, encryption type Triple DES cbc
      mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal ftp/thot.mds@THOTKB with kvno 3, encryption type DES cbc
      mode with CRC-32 added to keytab WRFILE:/etc/krb5.keytab.
```

La commande *ktadd* sert à extraire des *principals* du KDC vers un fichier de type *keytab*. Comme nous sommes ici dans le cas particulier où on vise le fichier *keytab* du KDC [2], on ne le donne pas en argument.

2. voir en 4.4 page 42 pour le cas général où le serveur telnet/FTP n'est pas le KDC

Configuration des utilisateurs Considérons le cheminement de l'authentification prévue pour un utilisateur. Celui-ci utilise *kinit* sur un système quelconque pour obtenir un TGT auprès du KDC *kerby.mds* de *KERBYKB.LOCAL*. Puis il demande, par exemple, à accéder au telnet de *thot.mds* dans *THOTKB*. Il demande donc à *kerby.mds* un TGT utilisable dans *THOTKB* : il reçoit *krbtgt/THOTKB@KERBYKB.LOCAL*. Il l'adresse alors au KDC de *THOTKB* (qui est *thot.mds*) qui l'accepte (car il fait confiance à *KERBYKB.LOCAL*) et retourne un ticket *host/thot.mds@THOTKB* (tous ces échanges de tickets sont illustrés page 25 et ne nécessitent pas d'entrer à nouveau de mot de passe).

Enfin l'utilisateur présente ce dernier ticket au serveur telnet, qui l'accepte, et va tenter d'ouvrir un *shell* au nom de l'utilisateur local *kerby* sur *thot.mds*. Dans cet état des choses, le telnet va échouer (en fait, il va demander le mot de passe de *kerby*) car l'utilisateur est authentifié dans *KERBYKB.LOCAL* et non dans *THOTKB* auquel appartient le service. Il faut donc que le compte local *kerby* contienne un fichier *~/.k5login* qui contienne la ligne *kerby@KERBYKB.LOCAL*. Ainsi, l'utilisateur sera autorisé sur le serveur. De plus, si des utilisateurs du domaine *THOTKB* souhaitent utiliser ce compte, ils devront également figurer dans ce fichier. Par exemple, si le contenu est

```
kerby@KERBYKB.LOCAL
kerby@THOTKB
```

alors les utilisateurs *kerby* des deux domaines Kerberos seront autorisés à utiliser le compte Solaris local *kerby* de *thot.mds*.

4.3.1 Les clients du MIT sous Solaris

Services MIT - Clients MIT

Le service telnet Dans ce test particulier, des utilisateurs loggés sur *thot.mds* (donc dans *THOTKB*) s'authentifient dans le domaine *KERBYKB.LOCAL* (dont le KDC est *kerby.mds*), puis reviennent dans *THOTKB* pour utiliser un service telnet. Ce cas de figure (peu probable) se produit ici dans la mesure où nous ne disposons que d'un système Solaris pour nos tests. Le résultat est donné en exemple 7 page 82.

On observe bien ici le mécanisme de confiance : le *kinit* permet d'obtenir le TGT (en saisissant un mot de passe) de *KERBYKB.LOCAL* (dont le nom est *krbtgt/KERBYKB.LOCAL@KERBYKB.LOCAL*, puis l'utilisateur veut utiliser une ressource en-dehors de son domaine. Il lui faut alors un TGT pour *THOTKB*, que *kerby.mds* accepte de lui donner (grâce au premier TGT). Enfin il le présente au KDC de *THOTKB* (le seul habilité à délivrer des tickets **@THOTKB*, et reçoit le ticket final *host/thot.mds@THOTKB* qui l'authentifie auprès du service telnet.

Le service FTP De même, tout se passe normalement pour les tranferts de fichiers. L'exécution de *ftp* permet l'acquisition du ticket *ftp/thot.mds@THOTKB*, ce qui suffit à s'authentifier auprès du service. Nous pouvons observer ce bon déroulement du test en exemple 7 page 87.

4.3.2 Les clients Heimdal sous Linux

Services MIT - Clients Heimdal

Le service telnet Le client telnet de Heimdal fonctionne toujours aussi bien avec le serveur du MIT, malgré l'authentification "croisée" entre les domaines.

La sortie écran est très proche de l'exemple 7 page 82.

Le service FTP De même, le FTP ne pose aucun problème. Nous constatons après-coups, comme d'habitude, que les tickets ont bien été acquis correctement auprès des deux KDC. Le résultat est similaire à celui de l'exemple 7 page 87.

4.3.3 Les clients KTelnet sous Windows

Services MIT - Clients KTelnet

Pour pouvoir utiliser ce logiciel dans notre configuration, il faut bien sûr ajouter le domaine *THOTKB* à la configuration, avec les mêmes informations que la section `[realms]` dans l'exemple 3 page 79. Puis nous définissons le domaine par défaut en *THOTKB* pour qu'il sache à quel KDC s'adresser lorsqu'il demande un ticket de service.

Le service telnet Nous initialisons l'authentification en demandant l'utilisateur *kerby* du domaine *KERBYKB.LOCAL*. Les tickets sont ensuite transmis comme d'habitude : un TGT pour *KERBYKB.LOCAL*, puis un pour *THOTKB* (délivré par *kerby.mds*), et un *host/thot.mds@THOTKB* obtenu de *thot.mds*. La session à l'écran est identique à l'exemple 6 page 81, où le service appartenait aussi au domaine *KERBYKB.LOCAL*.

On observe bien que ce service du domaine *THOTKB* accepte l'utilisateur *kerby* du domaine *KERBYKB.LOCAL* en lequel il a confiance.

Le service FTP Cette fonctionalité réussit tout aussi bien. Le ticket supplémentaire *ftp/thot.mds@THOTKB* est bien délivré par *thot.mds*.

4.4 Les services Heimdal sous Linux

Configuration des services Pour tester des services sous Heimdal faisant partie d'un domaine MIT Kerberos, nous devons créer les *principals* adéquats sur le KDC et les transférer sur le serveur Heimdal *amon.mds* :

```
root@thot:/# kadmin.local
kadmin.local:  ank -randkey host/amon.mds@THOTKB
WARNING: no policy specified for host/amon.mds@THOTKB; defaulting to no policy
Principal "host/amon.mds@THOTKB" created.
kadmin.local:  ank -randkey ftp/amon.mds@THOTKB
WARNING: no policy specified for ftp/amon.mds@THOTKB; defaulting to no policy
Principal "ftp/amon.mds@THOTKB" created.
kadmin.local:  ktadd -k /tmp/amon.keytab host/amon.mds@THOTKB ftp/amon.mds@THOTKB
Entry for principal host/amon.mds@THOTKB with kvno 4, encryption type Triple DES cbc
      mode with HMAC/sha1 added to keytab WRFILE:/tmp/amon.keytab.
Entry for principal host/amon.mds@THOTKB with kvno 4, encryption type DES cbc
      mode with CRC-32 added to keytab WRFILE:/tmp/amon.keytab.
Entry for principal ftp/amon.mds@THOTKB with kvno 3, encryption type Triple DES cbc
      mode with HMAC/sha1 added to keytab WRFILE:/tmp/amon.keytab.
Entry for principal ftp/amon.mds@THOTKB with kvno 3, encryption type DES cbc
      mode with CRC-32 added to keytab WRFILE:/tmp/amon.keytab.
```

Ceci a créé ces deux *principals*, et les a recopiés dans un fichier */tmp/amon.keytab* du même format que les fichiers *keytab* classiques. Puis, nous transférons ce fichier sur *amon.mds* et le renommons en */etc/krb5.keytab*.

Configuration des utilisateurs Comme en 4.3, les utilisateurs doivent créer et configurer des fichiers *˜/.k5login* (se référer à cette section pour plus de détails).

4.4.1 Les clients Heimdal sous Linux

Services Heimdal - Clients Heimdal

Naturellement, utiliser les clients Heimdal avec les services Heimdal ne pose pas de problème. Les demandes de tickets entre les deux domaines fonctionnent bien ; la session telnet figure en exemple 8 page 82 ; et la session FTP est proche de l'exemple 7 page 87.

4.4.2 Les clients du MIT sous Solaris

Services Heimdal - Clients MIT

De même, nous constatons bien l'interopérabilité entre le KDC du MIT, le telnet de Heimdal et le client du MIT ; la sortie obtenue est similaire à l'exemple 8 page 82. Le FTP est aussi satisfaisant, comme l'exemple 7 page 87.

4.4.3 Les clients KTelnet sous Windows

Services Heimdal - Clients KTelnet

Les deux clients (telnet et FTP) réussissent les mêmes tests que ceux de Heimdal et du MIT. Le client telnet continue de présenter quelques difficultés d'interprétation des premiers caractères du *shell*, mais cela ne gêne pas l'authentification ni la suite de la session. Comme d'habitude, la fenêtre de visualisation des tickets confirme que les requêtes se sont déroulées comme prévu.

4.5 Conclusion

Cette série de tests, suggérée par le document de Microsoft commenté au chapitre 1, se posait en alternative de la solution précédente. La différence est qu'ici, les ressources et les utilisateurs appartiennent à des domaines Kerberos différents (un du MIT, l'autre sous Windows). Les résultats des tests se sont montrés tout aussi concluants.

			Résultats	
KDC	Services	Clients	FTP	telnet
Authentification : Windows 2000 *Ressources* : MIT Solaris	MIT	MIT	√	√
		Heimdal	√	√
		KTelnet	√	√
	Heimdal	Heimdal	√	√
		MIT	√	√
		KTelnet	√	√

FIG. 4.1 - *Conclusions sur la relation de confiance unilatérale*

D'un point de vue fonctionnel, il n'y a donc pas vraiment d'intérêt de privilégier cette solution plutôt que la précédente. Celle-ci est juste un peu plus compliquée à mettre en œuvre (puisqu'il faut administrer deux KDC), mais reste praticable si on se trouve dans une situation qui s'y prête (un domaine de ressources déjà configuré avec un KDC du MIT, et un nouveau domaine d'utilisateurs sous Windows : il n'y a qu'à établir la relation de confiance).

Chapitre 5

KDC Solaris du MIT et Confiance Bilatérale

5.1 Rappel

Nous venons de voir au chapitre 4 comment établir une confiance unilatérale, où un domaine du MIT (sous Solaris 8) accepte les utilisateurs d'un domaine Windows 2000. Cette solution était proposée par Microsoft (au chapitre 1) dans le cas où des clients (sous UNIX ou Windows) souhaitent accéder à des ressources UNIX, en s'étant authentifiés auprès d'un KDC Windows.

Nous analysons ici le cas inverse : des utilisateurs s'authentifient auprès d'un KDC du MIT (sous Solaris 8) et souhaitent accéder à des ressources Windows. Microsoft conseille ici d'établir une relation de confiance bilatérale.

5.2 Etablir la confiance bilatérale

Ce chapitre complète le chapitre 4, en proposant de transformer la confiance unilatérale qu'il décrit en une relation bilatérale. Nous supposons donc maintenant être en présence des deux domaines KERBYKB.LOCAL *et* THOTKB *tels que nous les avons paramétrés en section 4.2 (page 38).*

L'état des domaines Nous décrivons à présent comment établir la confiance du domaine Windows *KERBYKB.LOCAL* en *THOTKB*, dont les KDC sont respectivement *kerby.mds* et *thot.mds*. Brièvement, rappelons que le paramétrage de la confiance unilatérale a consisté à déclarer le KDC de *THOTKB* auprès de Windows (à l'aide de la commande *ksetup*), puis à ajouter ce domaine du MIT à la liste des domaines Kerberos faisant confiance à *KERBYKB.LOCAL*, et enfin, sur *thot.mds*, à ajouter un *principal* correspondant au KDC Windows 2000 : *krbtgt/THOTKB@KERBYKB.LOCAL*.

Promotion en confiance bilatérale Pour promouvoir cette relation en confiance bilatérale, nous effectuons quasiment les mêmes opérations, en inversant les domaines. En effet, sur *kerby.mds*, il faut lui indiquer qu'il doit faire confiance au domaine *THOTKB*. Nous retournons à la console *Domaines et approbations*

Active Directory, sélectionnons les propriétés de *KERBYKB.LOCAL*, et choisissons l'onglet *Approbations*. Pour le moment, seule une confiance est paramétrée : celle que *THOTKB* a en *KERBYKB.LOCAL*[1]. Nous ajoutons donc *THOTKB* à la deuxième liste (*Domaines approuvés par ce domaine*), en spécifiant un mot de passe secret et en ignorant l'avertissement qui suit la validation (pour confirmer qu'il s'agit d'un domaine Kerberos, non-Windows, et interopérable).

Puis, sur *thot.mds*, nous ajoutons le *principal* qu'il utilisera pour délivrer des TGT destinés au domaine Windows :

```
root@thot:/# kadmin.local
Authenticating as principal kerby/admin@KERBYKB.LOCAL with password.
kadmin.local:  ank krbtgt/KERBYKB.LOCAL@THOTKB
WARNING: no policy specified for krbtgt/KERBYKB.LOCAL@THOTKB; defaulting to no policy
Enter password for principal "krbtgt/KERBYKB.LOCAL@THOTKB":
Re-enter password for principal "krbtgt/KERBYKB.LOCAL@THOTKB":
Principal "krbtgt/KERBYKB.LOCAL@THOTKB" created.
```

Le mot de passe tapé ici doit être le même que celui tapé ci-dessus sous Windows. Il peut bien sûr être différent de celui entré au chapitre 4 pour la confiance unilatérale. Il correspond à l'information que les deux KDC ont en commun pour garantir l'intégrité des tickets qu'ils vont s'échanger.

Résultat Malheureusement, cette manipulation ne conclut pas à un résultat satisfaisant. Nous avons suivi les instructions du document *Step-by-Step Guide to Kerberos 5 Interoperability* de Microsoft[2] : la confiance unilatérale configurée précédemment fonctionne toujours, mais la relation inverse (Windows faisant confiance au KDC du MIT) ne marche pas. En effet, un utilisateur du domaine *THOTKB* qui demande l'acquisition d'un ticket de service à *KERBYKB.LOCAL* ne reçoit qu'un message d'erreur :

```
root@thot:/# kinit kerby@THOTKB
Password for kerby@THOTKB:
root@thot:/# kvno host/amon.mds@KERBYKB.LOCAL
host/amon.mds@KERBYKB.LOCAL: KDC has no support for encryption type
                             while getting credentials
root@thot:/# klist -5
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: kerby@THOTKB

Valid starting     Expires            Service principal
02/21/02 15:40:33  02/22/02 01:40:33  krbtgt/THOTKB@THOTKB
02/21/02 15:40:41  02/22/02 01:40:33  krbtgt/KERBYKB.LOCAL@THOTKB
```

Ici, l'utilisateur *root* de *thot.mds* (qui est aussi le KDC de *THOTKB*) s'est authentifié en tant que *kerby@THOTKB* : l'exécution finale de *klist* et les logs de son KDC montrent que cette opération s'est bien déroulée (ce premier TGT est le premier des tickets listés ci-dessus). Puis nous demandons un ticket pour le service *host/amon.mds* (qui figure bien dans la base de données du KDC Windows, nous avons vérifié en nous authentifiant dans son domaine). Les

1. voir l'écran D.6 page 77
2. mentionné en annexe C

logs du KDC de *THOTKB* confirment que *kvno* lui a bien demandé le ticket *krbtgt/KERBYKB.LOCAL@THOTKB* (le deuxième ticket listé dans *klist*) pour pouvoir s'adresser au KDC de *KERBYKB.LOCAL*. Mais cette dernière requête échoue, avec le message affiché ci-desus. Signalons que *kvno* est un outil du MIT qui permet d'acquérir un ticket de service pour afficher sa version ; nous nous en sommes régulièrement servi pour nos tests préliminaires (simplement parcequ'il se charge de demander des tickets quelconques au KDC), avec succès. Nous avons effectué la même manipulation avec Heimdal sous Linux (où *kvno* est remplacé par *kgetcred*), avec le même résultat.

Sur le KDC Windows 2000 (le domaine de ressources auquel l'utilisateur a présente son deuxième TGT), le message d'erreur est le suivant :

```
Type de l'événement: Audit des échecs
Source de l'événement: Security
Catégorie de l'événement: Connexion de compte
ID de l'événement: 677
Date: 21/02/2002
Heure: 16:42:58
Utilisateur: AUTORITE NT\SYSTEM
Ordinateur: KERBY
Description:
Échec de la requête de ticket de service:
  Nom de l'utilisateur:
  Domaine de l'utilisateur:
  Nom du service: host/amon.mds
  Options de ticket: 0x0
  Code d'échec: 0xE
  Adresse du client: 172.16.8.136
```

On constate deux problèmes : le nom de l'utilisateur et son domaine (dans la requête de ticket) sont vides. Dans le cas des requêtes réussies, ces deux champs valent respectivement *kerby* et *KERBYKB.LOCAL* par exemple. Il semble donc que le TGT délivré par le KDC du MIT soit mal interprété par le KDC de Windows.

5.3 Conclusion

Il est regrettable de ne pas avoir pu mettre en œuvre le mécanisme de confiance bilatérale. Mais cet échec n'est pas d'une grande importance : en effet, nous avons déjà vu au chapitre 2 qu'il n'existe pas de couple kerbérisé service – client où le serveur soit sous Windows et le client sous UNIX [3].

Ainsi, cette configuration envisagée par Microsoft n'est en fait pas praticable (comme celle mentionnée dans la conclusion du chapitre 2). Leur document donne un exemple "concret" de cette configuration, où les utilisateurs accèdent à des partages de fichiers sous Windows : il est peu probable qu'ils l'aient réellement testé, puisqu'il n'existait pas de client SMB kerbérisé sous UNIX lors de la rédaction de leur document.

3. voir le paragraphe sur SAMBA à la page 27 pour notre tentative du partage de fichiers

Chapitre 6

KDC Solaris du MIT et Configuration des Clients

6.1 Rappel

Ce chapitre traite le cas où les authentifications et les ressources sont gérées par un KDC sous UNIX, en fait par l'implémentation du MIT sous Solaris 8. Le nom de cet unique domaine dans ce chapitre est *THOTKB*.

Nous avons vu au chapitre 1 que cette situation de *configuration des clients* s'applique aussi au cas où des clients Windows souhaitent accéder à des services UNIX lorsque le KDC est sous UNIX. Nous étendons ici les tests pour le cas où des clients UNIX souhaitent également accéder à ces ressources.

6.2 Les services du MIT sous Solaris 8

Nous n'avons pas eu à modifier la configuration du KDC du domaine *THOTKB* depuis nos tests précédents (chapitre 5). De même, les services telnet et FTP étaient déjà paramétrés (chapitre 4). Cet état de *THOTKB* qui approuve le domaine Windows *KERBYKB.LOCAL* n'empêche en rien les tests mentionnés ici. Plus précisemment, les résultats que nous abordons maintenant sont très proches de ceux du chapitre 4, car seule l'authentification du client change (elle a maintenant lieu dans le même domaine).

6.2.1 Les clients Heimdal sous Linux

Services MIT - Clients Heimdal

Le service telnet Nous avons commencé par tester ce service pour mettre en évidence les acquisitions de tickets, dont le TGT pour l'authentification. Dans l'exemple 9 page 82, nous pouvons bien constater la réussite de ce test (*thot.mds* implémente à la fois le KDC et le service), et observer le cache des tickets.

Le service FTP Puis nous testons les transferts de fichiers et terminons en listant les tickets acquis au cours des sessions telnet et FTP. On constate dans l'exemple 8 page 87 que tout s'est passé comme prévu : le cache contient le TGT et les deux tickets de service.

6.2.2 Les clients KTelnet sous Windows

Services MIT - Clients KTelnet

Ces clients sont bien compatibles avec cette configuration. La connexion telnet au client s'établit correctement, de la même manière qu'à l'exemple 6 page 81.

De même pour les transferts de fichiers en FTP, il n'y a pas de problème. Tout se déroule comme dans l'exemple 4 page 85 (mis à part les noms de machines et de domaines).

6.3 Les services Heimdal sous Linux

A nouveau, c'est *amon.mds* qui implémente les services Heimdal sous Linux. Nous n'avons pas eu à modifier sa configuration (par rapport au chapitre 4) car il continue d'appartenir au domaine Kerberos *THOTKB*.

6.3.1 Les clients Heimdal sous Linux

Services Heimdal - Clients Heimdal

Le service telnet Les clients MIT associés à leurs services ne posent à nouveau aucun problème (comportement identique à l'exemple 9 page 82). Notons également la bonne acquisition du TGT et du ticket de service dans *THOTKB*.

Le service FTP Quant à ce service, il reste bien sûr opérationnel. Le ticket *ftp/amon.mds@THOTKB* a bien été demandé au KDC (et accordé), comme dans le précédent exemple 8 page 87.

6.3.2 Les clients du MIT sous Solaris

Services Heimdal - Clients MIT

Le service telnet Pas non plus d'incompatibilité entre les clients du MIT et les services Heimdal. Dans l'exemple 10 page 83 nous exécutons la commande *whoami* pour illustrer le mapping entre le nom Kerberos *kerby@THOTKB* et l'utilisateur local *kerby* une fois la connexion telnet réussie. De même, le FTP du MIT interagit bien avec le service Heimdal : nos sessions de tests se comportent comme l'exemple 8 page 87.

6.3.3 Les clients KTelnet sous Windows

Services Heimdal - Clients KTelnet

Utiliser des clients KTelnet pour accéder à des services Heimdal en s'authentifiant auprès d'un KDC du MIT s'avère tout aussi concluant. Les connexions telnet et FTP s'établissent correctement, et les tickets sont bien acquis comme prévu.

6.4 Conclusion

Cette série de tests s'avère satisfaisante. Elle permet de conclure qu'il est possible de faire intéragir des implémentations différentes de Kerberos sous un KDC UNIX, simplement en adaptant les fichiers de configuration. Rappelons que l'implémentation SEAM (pour Solaris) a été retirée des tests suite à ses mauvais résultats au chapitre 3. Le tableau 6.1 (page 50) récapitule les tests réalisés ici.

			Résultats	
KDC	Services	Clients	FTP	telnet
MIT Solaris	MIT	Heimdal	√	√
		KTelnet	√	√
	Heimdal	Heimdal	√	√
		MIT	√	√
		KTelnet	√	√

FIG. 6.1 - *Conclusions sur les configurations de clients avec un KDC du MIT*

Chapitre 7

Mise en œuvre de l'authentification unique

Le Single Sign On Nous avons effectué un grand nombre de tests dans les chapitres précédents. Chacun d'eux s'est déroulé sous la forme d'une authentifications initiale (avec *kinit*) pour acquérir le TGT, puis des essais avec des clients (telnet et FTP). Ce mécanisme correspond au principe du **Single Sign On** : l'utilisateur utilise une seule fois son mot de passe et peut accéder à de multiples ressources.

Mais utiliser *kinit* dans ces conditions n'est pas très significatif. En effet, considérons les cas concrets suivants :

- L'utilisateur est physiquement devant la console [1] du système kerbérisé. L'interface (graphique ou texte) qu'il utilise pour s'y *logguer* doit donc directement utiliser Kerberos comme authentification : il obtiendra alors le TGT du domaine associé.

- L'utilisateur est *loggué* sur une autre machine hors du domaine (avec un nom et un mot de passe locaux), et il souhaite obtenir un *shell* sur un système kerbérisé : il doit donc utiliser *kinit* pour obtenir un TGT (le *Single Sign On*), puis utiliser un client de type telnet ou rlogin, et obtenir une copie de son TGT sur la machine distante.

Les questions abordées ici concernent les possibilités d'un *login* kerbérisé sur la console, et de "faire suivre" le TGT vers les hôtes contactés (*ticket forwarding*) pour éviter d'y ré-exécuter *kinit*.

Modification de krb5.conf Pour toutes nos authentifications en mode texte dans ce chapitre (*login* sur la console texte ou avec *kinit* à distance), il est nécessaire de configurer le comportement de la requête de TGT pour que celui-ci aie le *flag* `F` : ce ticket sera alors *forwardable* (nous parlerons plus bas de cette

1. Nous appelons "console" du système Solaris l'écran physiquement raccordé à la machine : son authentification est généralement *graphique* (grâce à l'écran de *login* apparaissant à la fin du démarrage du système), mais peut également s'effectuer sous la forme d'une ligne de commande (qui donne lieu à un *shell*). Ces deux modes utilisent des mécanismes d'authentification indépendants.

propriété). Pour cela, il faut éditer le fichier *krb5.conf*, et ajouter une ligne dans la section `[libdefaults]` :

```
[libdefaults]
    forwardable = true
```

Cela est indispensable pour obtenir des TGT *forwardables* lors d'un *login* texte sur la console, et permet d'omettre l'option *-f* lors de l'exécution distante de *kinit*.

7.1 Authentification sur la console

Configuration Nous nous plaçons ici dans le contexte suivant :

- Le KDC *kerby.mds* est sous Windows 2000, le nom du domaine est *KERBYKB.LOCAL*.
- La station de travail *thot.mds* est un système Solaris 8, utilisant l'implémentation Kerberos V5 du MIT ; l'utilisateur souhaite se *logguer* sur sa console. Cette machine est configurée pour appartenir au domaine mentionné ci-dessus.

Nous allons voir à présent deux solutions pour une authentification en ligne de commande (en remplaçant un fichier exécutable et avec un module PAM), puis une solution pour kerbériser l'authentification graphique.

Authentification en ligne de commande Les documents d'installation du MIT [2] spécifient qu'il faut remplacer */bin/login* par */usr/local/sbin/login.krb5*. Le fichier */etc/krb5.conf* est configuré comme celui de l'exemple 2 page 79.

Au démarrage du système, la console affiche le même écran que dans le *screenshot* de la page 74. Dans un premier temps, nous utilisons l'authentification en ligne de commande : nous choisissons donc *Command Line Login* dans les options : ceci utilise le fichier *login.krb5* que nous avons copié précédemment. Nous entrons alors le nom Kerberos et le mot de passe de l'utilisateur (ici *kerby*), et obtenons un *shell* local. En exécutant *klist*, nous constatons que le TGT a bien été obtenu auprès du KDC de *KERBYKB.LOCAL*.

```
thot console login: kerby
login: Password for kerby:
Last login: Tue Feb 19 21:18:04 from amon
Sun Microsystems Inc.   SunOS 5.8       Generic February 2000
kerby@thot:~$ klist -5 -f
Ticket cache: FILE:/tmp/krb5cc_p504
Default principal: kerby@KERBYKB.LOCAL

Valid starting     Expires            Service principal
02/19/02 22:28:54  02/20/02 08:28:54  krbtgt/KERBYKB.LOCAL@KERBYKB.LOCAL
        Flags: FIA
kerby@thot:~$
```

2. mentionnés dans les références en annexe C

L'option *-f* de *klist* permet d'afficher les *flags* des tickets. Nous constatons alors que le TGT possède les *flags* `I` (comme *Initial*: c'est le premier TGt acquis), `A` (qui signifie que le client a utilisé une pré-authentification), et `F` : celui-ci signifie que le TGT est *forwardable*. Après l'avoir essayé (en effectuant une connexion par telnet sur un autre système, par exemple), nous constatons que le TGT a effectivement été transmis [3] sur le système distant.

Sur le KDC *kerby.mds* sous Windows 2000, nous avons obtenu les logs suivants :

```
Type de l'événement: Audit des succès
Source de l'événement: Security
Catégorie de l'événement: Connexion de compte
ID de l'événement: 672
Date: 19/02/2002
Heure: 23:31:06
Utilisateur: AUTORITE NT\SYSTEM
Ordinateur: KERBY
Description:
Ticket d'authentification accordé:
  Nom de l'utilisateur: kerby
  Nom de domaine Kerberos fourni: KERBYKB.LOCAL
  ID de l'utilisateur: KERBYKB\kerby
  Nom du service: krbtgt
  No du service: KERBYKB\krbtgt
  Options du ticket: 0x40000000
  Type de cryptage du ticket: 0x1
  Type de pré-authentification: 2
  Adresse du client: 172.16.8.136
```

L'adresse IP mentionnée est bien celle de la station *thot.mds*. A l'issue de cette authentification locale, l'utilisateur dispose donc d'un TGT du domaine qu'il pourra utiliser pour accéder à ses ressources, et pourra le transmettre aux systèmes auquels il souhaitera se connecter.

Avec un module PAM ? La solution que nous venons de voir pour l'authentification en ligne de commande sur la console repose sur le remplacement du */bin/login* par une version du MIT. Or Solaris utilise plus habituellement les modules PAM pour ce mécanisme. Le paragraphe suivant (sur l'authentification graphique) explique en détails comment installer et paramétrer un tel module, il est ensuite facile d'adapter cette démarche pour une authentification en ligne de commande (en substituant *login* à *dtlogin* dans le fichier *pam.conf*). Cette solution, qui préserve le */bin/login* installé par Solaris, a l'avantage de rester compatible avec l'ensemble des mécanismes mis en œuvre par Sun autour de l'authentification, que le MIT n'a peut-être pas tous reproduits (cette remarque nous a été faite par Wyllys Ingersoll, travaillant chez Sun).

Authentification graphique Dans l'état actuel de la configuration, il est impossible de s'authentifier à l'aide de l'interface graphique illustrée page 74. Il s'avère qu'elle exploite les modules PAM paramétrés dans */etc/pam.conf* pour le

3. voir en 7.2 pour une bonne utilisation des tickets *forwardables* en telnet ou rlogin

service *dtlogin* (*Desktop login*). Nous téléchargeons donc un tel module utilisant Kerberos V5, à l'adresse `www.nectar.cc/krb/` (version 1.0.3), et le compilons (en indiquant le chemin vers nos fichier du Kerberos MIT au script *configure*, et en définissant la constante *LOG_DEBUG* dans les fichiers d'en-tête). Nous disposons donc maintenant d'un nouveau module PAM :

```
root@thot:/usr/lib/security# ls -l pam_krb5*
-rwxr-xr-x   1 root     other     696016 Feb 20 17:04 pam_krb5.so
lrwxrwxrwx   1 root     other         13 Feb 20 17:06 pam_krb5.so.1 -> ./pam_krb5.so
```

Nous activons ce mode d'authentification en modifiant (ajouter, commenter, décommenter des lignes) le fichier */etc/pam.conf*:

```
dtlogin   auth    optional  /usr/lib/security/pam_krb5.so.1 renewable forwardable
dtlogin   account optional  /usr/lib/security/pam_krb5.so.1 renewable forwardable
# dtlogin auth    required  /usr/lib/security/pam_unix.so.1 renewable forwardable
```

Nous tentons alors (après avoir redémarré l'écran de *login*) de nous authentifier. Après avoir entré le nom d'utilisateur Kerberos *kerby*, le système demande le mot de passe de *kerby@KERBYKB.LOCAL* : il l'accepte, et lance la session au nom de *kerby*. En exécutant *klist*, nous constatons que le TGT *krbtgt/KERBYKB.LOCAL* a bien été accordé par le KDC Windows 2000 (il figure dans ses logs), ainsi que le ticket *host/thot.mds*. Grâce aux mots-clefs `renewable forwardable` dans *pam.conf*, le TGT bénéficie des *flags* `FRIA`. Les `FIA` ont la même signification que dans le paragraphe précédent sur la console (donc le ticket est bien *forwardable*) ; le `R` signifie *renewable*, demandé dans *pam.conf*.

Conclusion sur la console En utilisant le binaire *login.krb5* pour la console texte et le module PAM pour la console graphique, nous réussissons bien à nous authentifier avec le nom Kerberos et donc à obtenir un TGT. Grâce à lui, nous pouvons bien obtenir d'autres tickets de services (*host*, *ftp*...). De plus, en ajoutant les mots-clefs mentionés plus haut aux fichiers *krb5.conf* et *pam.conf*, les TGT sont *forwardables*, et donc "suivront" l'utilisateur s'il se *loggue* sur un système distant.

7.2 Authentification distante

Configuration Plaçons-nous maintenant dans le cas où un utilisateur est déjà *loggué* sur une machine UNIX, et souhaite s'authentifier une fois auprès d'un KDC sous Windows 2000, pour pouvoir ensuite accéder aux ressources de ce domaine.

- L'utilisateur est *loggué* (de manière standard, sans Kerberos) sur un système Solaris *amon.mds*, dont l'implémentation Kerberos installée est celle du MIT. Il souhaite obtenir un *shell* sur le serveur *thot.mds*, qui fait partie du domaine *KERBYKB.LOCAL*.
- Le serveur *thot.mds* auquel l'utilisateur souhaite accéder : il dispose des services kerbérisés du MIT.

– Le KDC de ce domaine est *kerby.mds*, sous Windows 2000. Il devra donc délivrer le TGT au client, ainsi qu'un ticket *host/thot.mds*.

Les fichiers *krb5.conf* des systèmes UNIX sont à nouveau semblables à celui de l'exemple 2 page 79.

Authentification et telnet L'utilisateur doit, dans un premier temps, obtenir un TGT ; celui-ci est alors stocké sur sa machine cliente *amon.mds*. Puis il souhaite se connecter à *thot.mds* **et** y retrouver ce TGT. Cela s'appelle transmettre les tickets (*ticket forwarding*), et nécessite la modification du fichier *krb5.conf* mentionnée plus haut (ou alors l'utilisation de l'option *-f* avec *kinit*).

```
root@amon:/# kinit kerby
Password for kerby@KERBYKB.LOCAL:
root@amon:/# klist -5 -f
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: kerby@KERBYKB.LOCAL

Valid starting     Expires            Service principal
02/19/02 23:18:11  02/20/02 09:18:11  krbtgt/KERBYKB.LOCAL@KERBYKB.LOCAL
        Flags: FIA
root@amon:/# telnet -x -F -l kerby thot.mds
Trying 172.16.8.136...
Connected to thot.mds (172.16.8.136).
Escape character is '^]'.
Waiting for encryption to be negotiated...
[ Kerberos V5 accepts you as ``kerby@KERBYKB.LOCAL'' ]
[ Kerberos V5 accepted forwarded credentials ]
done.
Last login: Tue Feb 19 23:13:10 from amon
bash-2.03$ /usr/local/bin/klist -5 -f
Ticket cache: FILE:/tmp/krb5cc_p911
Default principal: kerby@KERBYKB.LOCAL

Valid starting     Expires            Service principal
02/19/02 23:21:20  02/20/02 09:18:11  krbtgt/KERBYKB.LOCAL@KERBYKB.LOCAL
        Flags: FfA
bash-2.03$
```

Notons que le *flag* `F` du TGT dans le premier *klist* (sur *amon.mds*) est positionné. Cela signifie qu'il est *forwardable*. Puis signalons que la ligne de commande du telnet contient l'option *-F*: cela demande au client telnet de transmettre une copie du TGT (ce qui est donc autorisé, vu son *flag* `F`), qui sera elle-même *re-forwardable* en cas de besoin (utiliser l'option *-f* à sa place pour *forwarder* un TGT non *re-forwardable*). Le dernier *klist*, exécuté sur le serveur *thot.mds*, illustre que le TGT a bien été transmis, et qu'il dispose à nouveau du *flag* `F` (*forwardable*), mais cette fois aussi du `f` (qui signifie que le TGT a été *forwardé*).

Authentification et rlogin Il est possible d'effectuer la même manipulation avec le client *rlogin* du MIT. Les significations des *flags* et des options restent

les mêmes.

```
root@amon:/# kinit kerby
Password for kerby@KERBYKB.LOCAL:
root@amon:/# rlogin -x -F -l kerby thot.mds
This rlogin session is using DES encryption for all data transmissions.
Last login: Tue Feb 19 23:27:40 from amon
bash-2.03$ /usr/local/bin/klist -5 -f
Ticket cache: FILE:/tmp/krb5cc_p973
Default principal: kerby@KERBYKB.LOCAL

Valid starting     Expires            Service principal
02/19/02 23:32:31  02/20/02 09:29:50  krbtgt/KERBYKB.LOCAL@KERBYKB.LOCAL
        Flags: FfA
bash-2.03$
```

A nouveau, tout se déroule comme prévu. Notons également, dans les deux cas, que les communications sont cryptées (option *-x* des commandes des clients).

Depuis la console Signalons que dans cette section, une fois l'étape du *kinit* passée, les exécutions de telnet et rlogin peuvent également s'effectuer depuis la console, une fois le TGT *forwardable* acquis [4] !

7.3 Résumé sur le Single Sign On

En terme d'authentification distante, nos essais se sont révélés concluants. En effet, grâce à la modification préliminaire du fichier *krb5.conf* (ou en spécifiant l'option *-f* à *kinit*), les TGT que nous obtenons sont *forwardables*; et en nous connectant à de nouveaux systèmes (telnet ou rlogin) avec l'option *-F*, nous transmettons donc ces TGT, qui sont eux-mêmes *forwardables* sur les systèmes destinations. Ainsi, en ayant entré une seule fois le mot de passe (lors du *kinit*), nous pouvons former une "chaîne" de *logins* de système en système, sans se ré-authentifier manuellement. Naturellement, cette chaîne peut contenir des systèmes de plusieurs domaines Kerberos distincts, reliés par des relations de confiance comme au chapitre 4.

Quant à l'authentification sur la console, nous avons bien trouvé deux mécanismes permettant l'obtention automatique d'un TGT (le nouveau */bin/login* et le module PAM), et celui-ci est bien *forwardable*.

Le mécanisme de *Single Sign On* est donc bien réalisable dans le cadre du protocole Kerberos.

4. la section 7.1 explique comment obtenir un tel TGT

Chapitre 8

Changer un mot de passe

8.1 Présentation

Protocole Le changement de mot de passe d'un utilisateur dans un domaine Kerberos correspond à l'utilisation d'un service kerbérisé standard. En effet, il existe un démon de changement de mot de passe, en général sur le KDC, qui attend des requêtes de clients. La démarche du client (*kpasswd* sous UNIX) est la même que pour tout autre service : acquisition d'un TGT, puis demande du ticket de service pour le changement de mot de passe : *kadmin/changepw@DOMAINE*. Rappelons qu'une fois un ticket de service acquis par le client, et une fois celui-ci authentifié auprès du service, ils disposent tous les deux d'une même clef de cryptage temporaire [1].

Grâce à cette clef de cryptage (assurant la confidentialité des échanges entre le client et le service de changement de mot de passe), le client va pouvoir communiquer son nouveau mot de passe, que le service inscrira dans la base de données du KDC. Quant au ticket de service acquis lors de cette opération, il est immédiatement détruit : en effet, il serait dangereux de conserver en cache un ticket qui permettrait si facilement d'altérer le point crucial de la sécurité Kerberos : le mot de passe d'un utilisateur.

Les tests Notre étude utilise deux types de KDC : Windows 2000, et celui du MIT. Pour tester leurs interopérabilités avec les différents clients, nous tentons des changements de mots de passe à l'aide des clients du MIT, de Heimdal, et de Windows.

Afin de pouvoir utiliser correctement les clients UNIX, il faut s'assurer de la présence d'une ligne `admin_server =` dans les descriptifs des domaines de la section `[realms]` de *krb5.conf* [2]. Ce sont ces serveurs que les clients contacteront pour transmettre les nouveaux mots de passe.

8.2 Avec un KDC sous Windows 2000

Dans un domaine Windows 2000, les utilisateurs sous Windows changent simplement leur mot de passe avec la combinaison de touches "*Ctrl-Alt-Del*".

1. c'était la conclusion de la section 2.3.3 page 16
2. voir les exemple de l'annexe E.1 page 78

Ceci est naturellement infaisable avec des clients UNIX, il existe donc des clients dédiés, de nom standard *kpasswd*.

8.2.1 Le client du MIT

Sur la station de travail Nous revenons donc dans la configuration où un utilisateur utilise un système Solaris 8 *thot.mds*, faisant partie du domaine *KERBYKB.LOCAL* de KDC *kerby.mds*. Il s'y authentifie, puis change son mot de passe :

```
root@thot:/etc# kinit kerby
Password for kerby@KERBYKB.LOCAL:
root@thot:/etc# kpasswd
Password for kerby@KERBYKB.LOCAL:
Enter new password:
Enter it again:
Password changed.
root@thot:/etc# klist -5
Ticket cache: FILE:/tmp/krb5cc_p543
Default principal: kerby@KERBYKB.LOCAL

Valid starting     Expires            Service principal
02/28/02 12:11:27  02/28/02 22:11:27  krbtgt/KERBYKB.LOCAL@KERBYKB.LOCAL
```

On constate que le client a validé le changement de mot de passe (après avoir demandé l'ancien, pour des raisons de sécurité), puis qu'aucun ticket (autre que le TGT) ne figure dans le cache. En retentant une exécution de *kinit*, nous nous assurons que le mot de passe a bien été changé. Il est donc possible de changer un mot de passe Windows depuis une station du MIT.

Sur le KDC Analysons maintenant les logs de Windows lors de cette opération. Une fois que l'utilisateur a entré son ancien mot de passe à *kpasswd*, deux événements apparaissent sur le KDC. Le premier est un échec pour une demande de ticket *kadmin/changepw*, et le second est un succès pour un nouveau TGT (ces deux tickets sont au nom de l'utilisateur *kerby*). Il est surprenant que Windows enregistre cela comme un échec, puisque la procédure réussit quand même. Puis, une fois le nouveau mot de passe tapé, aucun nouvel événement ne vient d'ajouter au journal du KDC.

Bien que cela ne rentre pas dans le cadre de l'étude d'interopérabilité, nous avons également utilisé le client du MIT pour changer un mot de passe sur un KDC du MIT : celui-ci enregistre la délivrance du ticket *kadmin/changepw* comme un succès (dans ses logs). Il est donc surprenant que Windows gère cette opération de changement de mot de passe comme une erreur.

Il nous aurait semblé logique que le KDC génère une nouvelle clef de session (entre lui-même et l'utilisateur) puisque le mot de passe a changé, il la transmettrait donc au client dans un nouveau TGT [3]. Pourtant, après vérifification, il s'avère que le TGT listé dans le *klist* est celui qui avait été délivré par le *kinit* et non par le *kpasswd* (comparaison des dates d'émission). Ce nouveau TGT n'est donc pas non plus enregistré dans le cache [4].

3. voir en 2.1.1 page 11 (et le schéma de la page 12) pour la construction d'un TGT
4. voir la remarque sur ce client page 61 pour une tentative d'explication

8.2.2 Le client Heimdal

Nous essayons à présent, depuis un système Heimdal *verone* sous Linux, de changer un mot de passe utilisateur sur un KDC Windows 2000. Malheureusement, le client Heimdal semble bien moins compatible que celui du MIT :

```
verone:~# kinit kerby
kerby@KERBYKB.LOCAL's Password:
verone:~# kpasswd
kerby@KERBYKB.LOCAL's Password:
New password:
Verifying password - New password:
kpasswd: krb5_change_password: Message out of order
verone:~# klist
Credentials cache: FILE:/tmp/krb5cc_0
        Principal: kerby@KERBYKB.LOCAL

  Issued           Expires          Principal
Feb 28 15:50:02  Mar  1 01:48:45  krbtgt/KERBYKB.LOCAL@KERBYKB.LOCAL
```

Le message d'erreur du client (`Message out of order`) semble être un problème de compatibilité entre les deux implémentations de Kerberos. Sur le KDC, les logs sont exactement les mêmes que dans la section précédente : un échec pour *kadmin/changepw* et une réussite pour un nouveau TGT, qui n'apparaît pas dans le cache. L'implémentation de Heimdal semble donc différente du standard suivi par le MIT et Microsoft.

8.3 Avec un KDC du MIT

Nous utilisons à nouveau *thot.mds* comme KDC du domaine *THOTKB*. Pour tester son interopérabilité dans le cadre des changements de mots de passe, nous essayons les clients Heimdal et Windows.

8.3.1 Le client Heimdal

Nous reconfigurons *verone* pour s'adresser au domaine *THOTKB*, et tentons à nouveau l'exécution de *kpasswd* :

```
verone:~# kinit kerby
kerby@THOTKB's Password:
verone:~# kpasswd
kerby@THOTKB's Password:
New password:
Verifying password - New password:
kpasswd: krb5_change_password: Message out of order
```

Il s'agit bien des mêmes messages que dans le cas du KDC sous Windows 2000. Cela conforte notre idée d'incompatibilité entre Heimdal et les implémentations de référence du MIT et de Microsoft. Le KDC a généré une seule ligne de log, après que l'utilisateur ait entré l'ancien mot de passe :

```
AS_REQ (2 etypes {1 3}) 172.20.0.2(88): ISSUE: authtime 1014910541,
       etypes {rep=1 tkt=16 ses=1}, kerby@THOTKB for kadmin/changepw@THOTKB
```

Il s'agit du même ticket de service *kadmin/changepw* que dans le cas du premier KDC. Mais ici, aucun nouveau TGT n'est délivré (ce n'est pas non plus le cas lorsque nous utilisons le client du MIT avec succès, donc ce n'est pas un problème).

8.3.2 Windows

Ce cas correspond à utiliser un client Windows pour changer un mot de passe Kerberos sur un KDC du MIT sous Solaris. Or la manière standard pour changer son mot de passe sous Windows correspond à utiliser la combinaison de touches "*Ctrl-Alt-Del*" puis choisir l'action correspondante. Cela signifierait donc que la station de travail Windows appartienne au domaine Kerberos comme s'il s'agissait d'un domaine Windows 2000. Or les solutions de Microsoft [5] pour les KDC sous UNIX avec des clients sous Windows étaient :

Confiance bilatérale Les utilisateurs s'authentifient sur un KDC Windows, et doivent utiliser des ressources d'un KDC sous UNIX [6]. Pour changer de mot de passe, un utilisateur s'adresse donc à son KDC sous Windows, normalement.

Configuration de client Les utilisateurs s'authentifient auprès d'un KDC sous UNIX pour accéder à des ressources sous UNIX également [7]. C'est dans ce cas de figure que nous nous plaçons donc ici.

Dans cette deuxième configuration, nous avons effectué nos tests sur les services telnet et FTP du MIT et de Heimdal, avec des clients particuliers pour Windows [8]. Nous cherchons donc également des clients de type *kpasswd* pour Windows.

KTelnet Ce programme, déjà utilisé pour le telnet et le FTP, propose également un client de changement de mot de passe. Nous nous authentifions donc en tant que *kerby@THOTKB* (ce qui nous permet d'acquérir le TGT *krbtgt/THOTKB@THOTKB*), puis tentons de changer de mot de passe (ce qui se fait par l'intermédiaire du *Ticket Manager*). Malheureusement, nous obtenons également une erreur. Le client indique *Server replies: Failed reading application request* (on dispose toujours du même TGT, sans nouveau ticket) et le KDC a indiqué qu'il a délivré le ticket *kadmin/changepw@THOTKB* au client *kerby@THOTKB*. Le mot de passe n'a bien sûr pas été changé. Il s'agit donc probablement de la même erreur de compatibilité qu'entre Heimdal et le KDC du MIT [9].

Leash32 du MIT Il s'agit d'un outil sous Windows téléchargeable sur le site du MIT [10]. Il offre une interface graphique pour la gestion de tickets provenant

5. au chapitre 1 de cette partie
6. ce cas a été vu au chapitre 5, et était peu concluant
7. ce cas a été traité au chapitre 6
8. KTelnet, introduit en 3.1.2 page 20 et testé en 6.2.2 (page 49) et 6.3.3 (page 49)
9. il semblerait que l'auteur de KTelnet pour Windows ait utilisé Heimdal comme référence
10. comme les autres fichiers mentionnés en 3.2.3 page 22

d'un KDC du MIT. Bizarrement, cet outil (version 2.1.1.1) génère systématiquement des erreurs lors de chaque échange avec le KDC : il essaie les protocoles Kerberos V4 et V5, or nous n'utilisons que le V5. Bien que cela n'empêche pas l'acquisition d'un TGT pour Kerberos V5, nous obtenons une erreur (du protocole V4) qui arrête la procédure lorsque nous tentons un changement de mot de passe.

8.4 Résumé

Peu de ces tests se sont donc avérés concluants. Le tableau 8.1 récapitule les tests que nous avons effectués, y compris celui où un client du MIT dialogue avec un KDC du MIT : nous ne l'avons pas détaillé ici, mais il nous a servi pour s'assurer des logs auquels nous devions nous attendre pour les autres tests.

KDC	Clients	Résultats
Windows 2000	Heimdal	X
	MIT	√
MIT	Heimdal	X
	Windows	X
	(MIT)	(√)

FIG. 8.1 - *Conclusions sur les changements de mots de passe*

Dans l'ensemble, Heimdal est incompatible avec les autres implémentations, et il n'y a pas de client Windows disponible pour interopérer avec un KDC du MIT.

Remarque sur le client du MIT Nous avons pensé à une explication sur le comportement du client du MIT avec un KDC Windows. Comme stipulé dans les normes Kerberos, celui-ci tente d'utiliser le service *kadmin/changepw*, qui n'existerait simplement pas sous Windows 2000. Il est possible que l'échec signalé par le serveur permette au client de supposer qu'il s'agisse d'un KDC Windows, et qu'il utiliserait alors une procédure propre à ce cas de figure : utiliser un nouveau TGT (voir en 8.2.1). Le client Heimdal respecte peut-être strictement la norme Kerberos, et n'aurait donc pas intégré cette procédure de changement de mot de passe utilisée par Windows. Cela expliquerait les meilleurs résultats du MIT.

Conclusion

Notre travail

Méthodologie Nous avons donc achevé les tests que nous avions prévu de faire. La marche à suivre, indiquée par Microsoft au chapitre 1, nous a semblé raisonable et a effectivement conduit à de bons résultats. En ayant mis en œuvre des mécanismes aussi différents que les simples configurations de clients, les confiances (unilatérale et bilatérale) entre domaines, et les comptes de services, nous avons eu l'occasion de manipuler de nombreuses notions de Kerberos, et d'entamer de nombreuses discussions (*newsgroups* et emails) avec d'autres utilisateurs réguliers de ce protocole (certains affiliés au MIT ou à Sun).

Résultats La plupart de nos tests se sont avérés positifs. Comme le récapitule le tableau 9.2 page 65, il existe de nombreuses combinaisons d'implémentations et de solutions pour faire interagir différents intervenants de Kerberos. Nous sommes arrivés à un certain nombre de conclusions.

- Il n'existe pas de client UNIX capable d'utiliser des services kerbérisés de Windows 2000. La solution de *configuration de clients* pour le cas d'un KDC sous Windows est donc intestable.
- Utiliser un unique KDC sous Windows pour les utilisateurs et des ressources UNIX est parfaitement praticable avec la solution des *comptes de services*.
- Une solution alternative est la *confiance unilatérale* entre un domaine de ressources UNIX (dont le KDC est implémenté par le MIT) et un domaine d'utilisateurs Windows. Celle-ci nous a paru tout aussi efficace que la précédente, bien que moins évidente à mettre en œuvre et à administrer (puisqu'il y a deux domaines distincts à gérer).
- La solution de *confiance bilatérale* (préconisée dans le cas où des utilisateurs d'un domaine UNIX souhaitent accéder à des ressources d'un domaine Windows) nous paraît à nouveau impraticable. En effet, en plus du manque de services Windows kerbérisés proposant des clients UNIX, nous nous sommes heurtés à une difficulté de configuration de la confiance bilatérale. Bien que probablement réalisable, le manque de services à tester laisse cette solution dans le domaine du théorique.
- Quant à utiliser des services de type UNIX en s'authentifiant auprès d'un KDC UNIX (avec des clients quelconques), cela s'est avéré très praticable.

Cette solution de simple *configuration des clients* est préconisée dans le cas, par exemple, de la migration d'un domaine NIS vers Kerberos.

- Kerberos correspond bien au principe du *Single Sign On* : en s'authentifiant une fois pour toutes avec un mot de passe (physiquement sur une console ou à distance avec *kinit*), il est facile de s'authentifier automatiquement de système en système en "emmenant" ses TGT avec soi.

- Il reste le problème du changement de mot de passe. Heimdal semble incompatible avec Windows 2000 et le KDC du MIT, et nous manquons de clients efficaces sous Windows pour changer un mot de passe auprès d'un KDC du MIT.

En conclusion sur l'interopérabilité des différents systèmes Kerberos étudiés, nous considérons qu'en configurant correctement les systèmes d'un domaine (notamment au sujet des types de cryptage supportés), il est tout à fait possible de mélanger des implémentations différentes. Concernant les interactions entre Windows et les systèmes UNIX, il faut garder à l'esprit que ces deux univers utilisent des ressources complètement différentes (de type telnet - FTP d'un côté, partages de fichiers NetBIOS de l'autre), et posent avant tout des problèmes de disponibilité de clients pour les différents services.

Et après ?

Au fil des tests, un certain nombre d'idées nous sont venues, qui pourraient encore étayer la palette de résultats fournis dans ce document. Nous les listons ici, pour une éventuelle reprise ultérieure de cette étude.

- Reprendre les tests des chapitres 4, 5 et 6 en utilisant un KDC implémenté par Heimdal plutôt que par le MIT. En effet, nous avons choisi de nous concentrer sur un nombre "réduit" de tests en ométant cette solution pour offrir plus de détails dans nos résultats. Il serait maintenant intéressant de reprendre les tests de ces chapitres avec un KDC sous Linux utilisant Heimdal.

- Tester des propagations des mots de passe entre des KDC *master* et *slave* hétérogènes. Par exemple, configurer un contrôleur primaire de domaine Windows 2000 et un KDC *slave* du MIT (ou même de Heimdal) et tester l'authentification des utilisateurs auprès de ce serveur secondaire.

- Ajouter au chapitre 7 (sur le *Single Sign On*) un descriptif de la procédure à suivre pour pouvoir se *logguer* sur la console d'une station Linux/Heimdal aboutissant à l'acquisition d'un TGT par un KDC Windows 2000 ou du MIT (il s'agirait d'une section semblable à la 7.1 où nous décrivons l'authentification sur une console Solaris/MIT).

- Tenter la configuration où une station de travail Windows 2000 serait configurée comme appartenant à un domaine Kerberos implémenté par un KDC du MIT, et le client qui s'y logguerait tenterait d'utiliser des ressources Windows (partages de fichiers).

– Résoudre notre problème d'utilisation de SAMBA [11] avec des versions futures de SAMBA ou de OpenLDAP ou du Kerberos du MIT. En réussissant la compilation d'un client fonctionnel, il deviendrait possible d'effectuer des tests aux chapitres 2 et 5 (où des clients UNIX pourraient accéder à des partages de fichiers kerbérisés) ; et si nous pouvions disposer d'un serveur SAMBA kerbérisé compatible avec Windows 2000/XP, nous pourrions ajouter de nouveaux tests aux chapitres 3, 4 et 6 (où des clients Windows pourraient accéder à des partages de fichiers sur des serveurs UNIX).

– Réussir malgré tout les changements de mots de passe qui ont échoué au chapitre 8. Cela revient à faire marcher le *kpasswd* de Heimdal avec un KDC du MIT ou un serveur Windows 2000 (peut-être en essayant une version de dévelopement ?), et à trouver un outil sous Windows qui pourrait changer un mot de passe sur un KDC du MIT.

– Utiliser *gsscred* (dans */usr/sbin*) pour éviter de créer des fichier *~/.k5login*. Cet exécutable sert à créer des *mappings* entre les *principals* Kerberos et les UID locaux du système UNIX : les utilisateurs n'ont plus à maintenir ces fichiers de configuration, c'est l'administrateur qui met en place une politique de *mapping*, utilisable dans le cas des confiances entre domaines [12].

– Toujours pour tenter de simplifier la reconnaissance des *principals* d'un domaine Kerberos à l'autre, Douglas E. Engert nous a proposé un *patch* à appliquer au fichier *an_to_ln.c* du Kerberos V5 1.2.3 du MIT [13]. Celui-ci permet de créer un *mapping* implicite entre les utilisateurs d'un domaine Kerberos de confiance et notre propre domaine, en l'absence de fichier *~/.k5login*. Autrement dit, dans le cas où un utilisateur d'un domaine de confiance vient se *logguer* dans notre domaine : s'il existe un fichier *~/.k5login*, rien ne change ; s'il n'en existe pas, l'autorisation est accordée. Ce *patch*, à essayer, figure en annexe H.

11. décrit au chapitre 2 page 27
12. voir notre utilisation des fichiers *~/.k5login* en 4.3 au paragraphe *Configuration des utilisateurs* page 41
13. depuis nos tests, la version 1.2.4 est parue au MIT, mais nous n'avons pas eu le temps d'en tenir compte dans nos démarches

KDC d'authentification	Serveurs	Clients	Solutions	Réussites	pages
Windows 2000	Heimdal	Heimdal	Comptes de services	telnet, FTP	32
			Confiance unilatérale	telnet, FTP	43
		MIT	Comptes de services	telnet, FTP	33
			Confiance unilatérale	telnet, FTP	43
		KTelnet	Comptes de services	telnet, FTP	33
			Confiance unilatérale	telnet, FTP	43
		SEAM	Comptes de services	telnet, FTP	32
	SEAM	SEAM	Comptes de services	FTP	34
	MIT	Heimdal	Comptes de services	telnet, FTP	36
			Confiance unilatérale	telnet, FTP	41
		MIT	Comptes de services	telnet, FTP	36
			Confiance unilatérale	telnet, FTP	41
		KTelnet	Comptes de services	telnet, FTP	36
			Confiance unilatérale	telnet, FTP	42
	mot de passe	MIT	kpasswd		58
	Single Sign On avec MIT	console	graphique et texte		52
		distant	avec *kinit*		54
MIT sous Solaris 8	MIT	Heimdal	Configuration de clients	telnet, FTP	48
		KTelnet	Configuration de clients	telnet, FTP	49
	Heimdal	Heimdal	Configuration de clients	telnet, FTP	49
		MIT	Configuration de clients	telnet, FTP	49
		KTelnet	Configuration de clients	telnet, FTP	49

FIG. 9.2 - *Conclusions générales (tests réussis)*

Annexes

Annexe A

Glossaire

Authentication Service (AS) C'est un des deux services rendus par le KDC. Un client s'adresse à lui pour obtenir un TGT, qu'il pourra alors présenter au TGS pour recevoir des tickets de services.

Authentificateur C'est un bloc de données envoyé par un client vers un service standard ou un TGS. Etant crypté avec une clef de session, il permet au service de s'assurer de l'identité de l'émetteur. Bénéficiant d'un mécanisme de *timestamp*, il ne peut être utilisé deux fois, contrairement à un ticket. Cet élément du protocole est commenté dans la première partie, pour expliquer son fonctionnement. Mais nous ne le mentionnons pas dans nos tests de la seconde partie : son envoi est implicite à chaque fois que nous parlons de l'émission d'un ticket (TGT ou ticket de service).

Cryptage symétrique Il s'agit d'un mode de cryptage où la même clef sert au cryptage et au décryptage. La difficulté est donc de transmettre la clef aux deux partis en la gardant secrète des intrus. Ce terme s'oppose au cryptage asymétrique, où ces deux opérations s'effectuent à l'aide d'une paire de clefs distinctes en relation mathématique.

Domaine Nous nommons ainsi les *domaines Kerberos*. Dans les documents en Anglais, le mot *domain* est couramment utilisé pour les domaines Windows (qui sont des domaines Kerberos dans le cas de Windows 2000), et le mot *realm* correspond aux domaines Kerberos en général, mais surtout sous UNIX.

Generic Security Services Application Programming Interface (GSSAPI) C'est une API définissant les mécanismes de sécurité Kerberos offerts aux programmeurs sous UNIX (son équivalent sous Windows est la SSPI : *Security Service Provider Interface*). Cet acronyme apparaît régulièrement dans les logs de l'annexe E puisqu'elle est utilisée dans tous les serveurs et clients kerbérisés.

Kerberos C'est le protocole d'authentification dont nous testons l'interopérabilité. Il est issu du MIT ; son nom vient de l'appellation grecque du chien à trois têtes *Cerbère*. Cet animal mythologique étant lui-même un gardien (du royaume des morts *Hadès*, en l'occurence), il a donné son nom à ce

protocole dont le rôle est de garantir la sécurité et l'intégrité des données qui circulent sur le réseau.

Key Distribution Center (KDC) C'est le processus qui implémente les services d'authentification (AS) et de délivrance de tickets (TGS). Il est possible de distinguer ces deux services en les répartissant sur des machines différentes.

Massachusetts Institute of Technology (MIT) C'est l'organisme à l'origine du protocole Kerberos, développé dans le cadre du projet Athena (un réseau informatique d'étudiants à très grande échelle).

Mot de passe Une chaîne de caractères associée à un *principal*. Il est supposé secret, connu seulement de l'utilisateur ou du service auquel il est associé, et du KDC (qui les connaît tous). On considère qu'il est possible de (dé)crypter des informations en utilisant un mot de passe comme clef : on applique en réalité une fonction de hachage sur le mot de passe pour en extraire une clef numérique.

NT LAN Manager (NTLM) C'est le protocole d'authentification utilisé sous Windows NT jusqu'à la version 4. Bien moins sécurisé que Kerberos V5, il reste disponible sous Windows 2000 pour des raisons de compatibilité.

Pluggable Authentication Module (PAM) Il consiste en une bibliothèque contenant des services d'authentification, gestion de comptes... qu'un système peut utiliser pour ses différents services nécessitant une authentification. Nous avons vu en 7.1 (dans la partie du *login* graphique, page 53) comment nous avons utilisé un tel module pour une authentification kerbérisée sous Solaris 8.

Pré-authentification Il est fréquent, dans le protocole Kerberos, d'effectuer une pré-authentification lors de la demande d'un TGT. Cela signifie que le client crypte un *timestamp* avec le mot de passe ; et le KDC le vérifie. Si le test échoue, c'est qu'il s'agit d'un pirate qui souhaite obtenir une copie d'un TGT : on ne le lui envoie donc pas, et il ne pourra pas effectuer d'attaque à dictionnaire pour trouver le mot de passe.

Principal Un *principal* ("mandant" en français), dans le cadre du protocole Kerberos, est une chaîne de caractères qui correspond à un intervenant dans un échange d'informations sécurisées. Il s'agit donc ici des utilisateurs et des services kerbérisés. Voir en 1.4 page 10 pour les conventions de nommage. Chaque *principal* dispose d'un mot de passe secret, connu de lui-même et du KDC.

SAMBA Il s'agit d'une implémentation libre du protocole SMB (*Server Message Block*). C'est celui utilisé nativement sous Windows pour partager les fichiers et les imprimantes des serveurs et postes de travail (accessibles par le voisinage réseau). Cette implémentation est disponible sur le site `www.samba.org` ; elle utilise essentiellement le protocole d'authentification NTLM, mais supporte depuis peu (version 3.0, par CVS), expérimentalement, la version 5 de Kerberos [1].

1. voir le chapitre 2 page 27 à ce sujet

Single Sign On (SSO) C'est le nom donné à tout mécanisme permettant à l'utilisateur de s'authentifier une seule fois (par mot de passe), et d'accéder ensuite à diverses ressources grâce à cette pré-authentification. Dans le cas de Kerberos, la possibilité de transmettre les TGT d'un système à l'autre correspond bien à ce mécanisme, et est expliqué au chapitre 7 page 51.

Ticket C'est un sensemble d'informations destinées à un service, permettant d'attester qu'un utilisateur s'est authentifié auprès du KDC. Cet ensemble est crypté à l'aide du mot de passe du service ciblé.

Ticket-Granting Service (TGS) C'est un des services rendus par le KDC. Un utilisateur s'adresse à lui en présentant un TGT, pour obtenir un ticket dédié à un service cible.

Ticket-Granting Ticket (TGT) C'est un *ticket de délivrance de ticket*. Il est délivré à un utilisateur par le service AS du KDC lors de son authentification. On lé présente au service TGS pour obtenir des tickets dédiés à des services cibles.

Annexe B

Moyens techniques

Pour pouvoir utiliser facilement plusieurs systèmes d'exploitation (Windows 2000 Server, Windows XP, Linux Debian, Sun Solaris...) avec peu de PC, nous avons eu recours à un logiciel : *VMware Workstation*.

Il s'agit d'un outil (que nous avons utilisé sous sa version Windows) qui simule un PC complet : carte mère, disques durs, disquettes, lecteurs CD, ports série et parallèle, bus USB, carte vidéo, cartes réseau...

Ainsi, à l'aide de *bridges* réseaux (implémentés par nos PC réels avec Windows XP), nous avons pu mettre les différents intervenants Kerberos en interaction sur les systèmes d'exploitation de notre choix.

Pour plus d'informations : `www.vmware.com`

Annexe C

Références

Certaines des sources d'informations que nous listons ici sont mentionées dans notre rapport à plusieurs endroits, alors que d'autres ne nous ont servi qu'à nous guider, sans être cités dans ce document.

Designing an Authentication System: a Dialogue in Four Scenes Ce (long) dialogue fictif du MIT, entre deux informaticiens, explique pas à pas toutes les fonctionalités de base du protocole Kerberos V4, et leurs raisons d'être : ils commencent par décrire un protocole très basic, et le renforcent de jour en jour. La fin de ce document explique les principales différences entre les version 4 et 5 de Kerberos.
⇒ `web.mit.edu/kerberos/www/dialogue.html`

Kerberos V5 Installation Guide C'est un guide d'installation pour l'implémentation du MIT. Même si on utilise d'autres implémentations, ce guide est très utile dans la mesure où les instructions qui s'y trouvent servent de référence pour tous. Il existe aussi deux autres fichiers, pour l'administration et l'utilisation.
⇒ `web.mit.edu/kerberos/www/krb5-1.2/krb5-1.2.3/doc/install-guide.ps` et `admin-guide.ps` et `user-guide.ps`

Windows 2000 Kerberos Interoperability Ce document de Microsoft fait l'objet du chapitre 1 page 24. Il est à l'origine de la classification de nos tests selon les différentes solutions d'interopérabilité disponibles.
⇒ `www.microsoft.com/windows2000/techinfo/howitworks/security/kerbint.asp`

Kerberos Network Security in the Solaris Operating Environment Ce document PDF explique pas à pas comment configurer le service Kerberos sous Solaris, en utilisant l'implémentation de SUN : SEAM.
⇒ `www.sun.com/blueprints/1001/krb.pdf`

Step-by-Step Guide to Kerberos 5 Interoperability C'est une page HTML de Microsoft qui décrit un ensemble de procédures pour faciliter la configurations de domaines Kerberos hétérogènes. Nous nous en sommes parfois servi pour mettre au point certaines maquettes de tests.
⇒ `www.microsoft.com/windows2000/techinfo/planning/security/kerbsteps.asp`

Le Projet pam_krb5_module de J. A. Vidrine C'est le module PAM que nous avons téléchargé et compilé sous Solaris, pour tester le *Single Sign On* à l'aide de l'écran de *login* graphique.

⇒ `www.nectar.cc/krb/`

Heimdal sous Linux C'est une implémentation libre du protocole Kerberos V5. Nous nous en sommes servis pour nos tests sur plateforme Linux Debian. La procédure d'installation est décrite en 3.2.1 page 21.

⇒ `www.pdc.kth.se/heimdal`

SEAM sous Solaris 8 C'est une implémentation propriétaire de Kerberos V5, développée par Sun. Nous l'introduisons dans le rapport en 3.2.2 page 21, et expliquons en 3.5 page 36 pourquoi nous avons décidé de l'abandonner.

⇒ `www.sun.com/bigadmin/content/adminPack/`

Kerberos V5 du MIT C'est l'implémentation développée par le MIT, qui est à l'origine du projet Kerberos. Elle fait donc référence en matière d'interopérabilité. Nous l'avons installé sous Solaris pour remplacer SEAM qui ne nous convenait pas (cette implémentation est présentée en 3.2.3 page 22).

⇒ `web.mit.edu/kerberos`

KTelnet sous Windows C'est une implémentation de clients telnet et FTP kerbérisés, indépendante de Microsoft. Introduit dans ce document en 3.1.2 page 20, son site Internet propose également une documentation en PDF.

⇒ `www.stacken.kth.se/~thn/ktelnet`

Annexe D

Copies d'écran

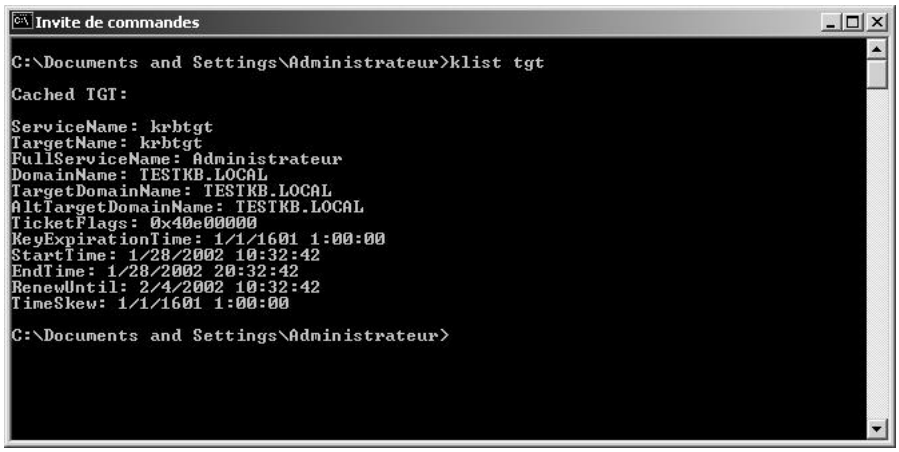

FIG. D.1 - *Exemple d'écran KList*

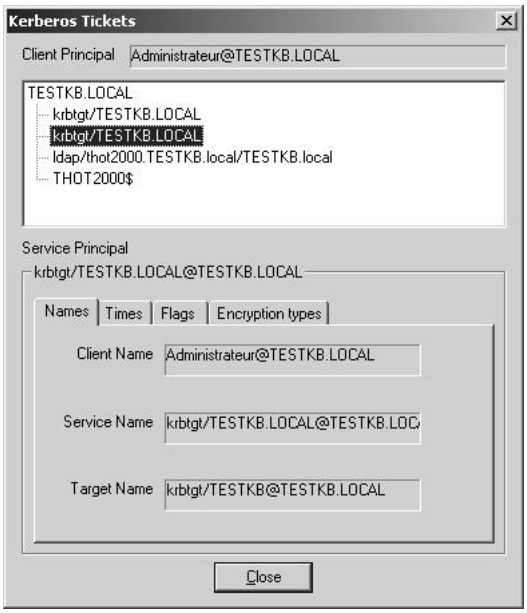

FIG. D.2 - *Exemple d'écran KerbTray*

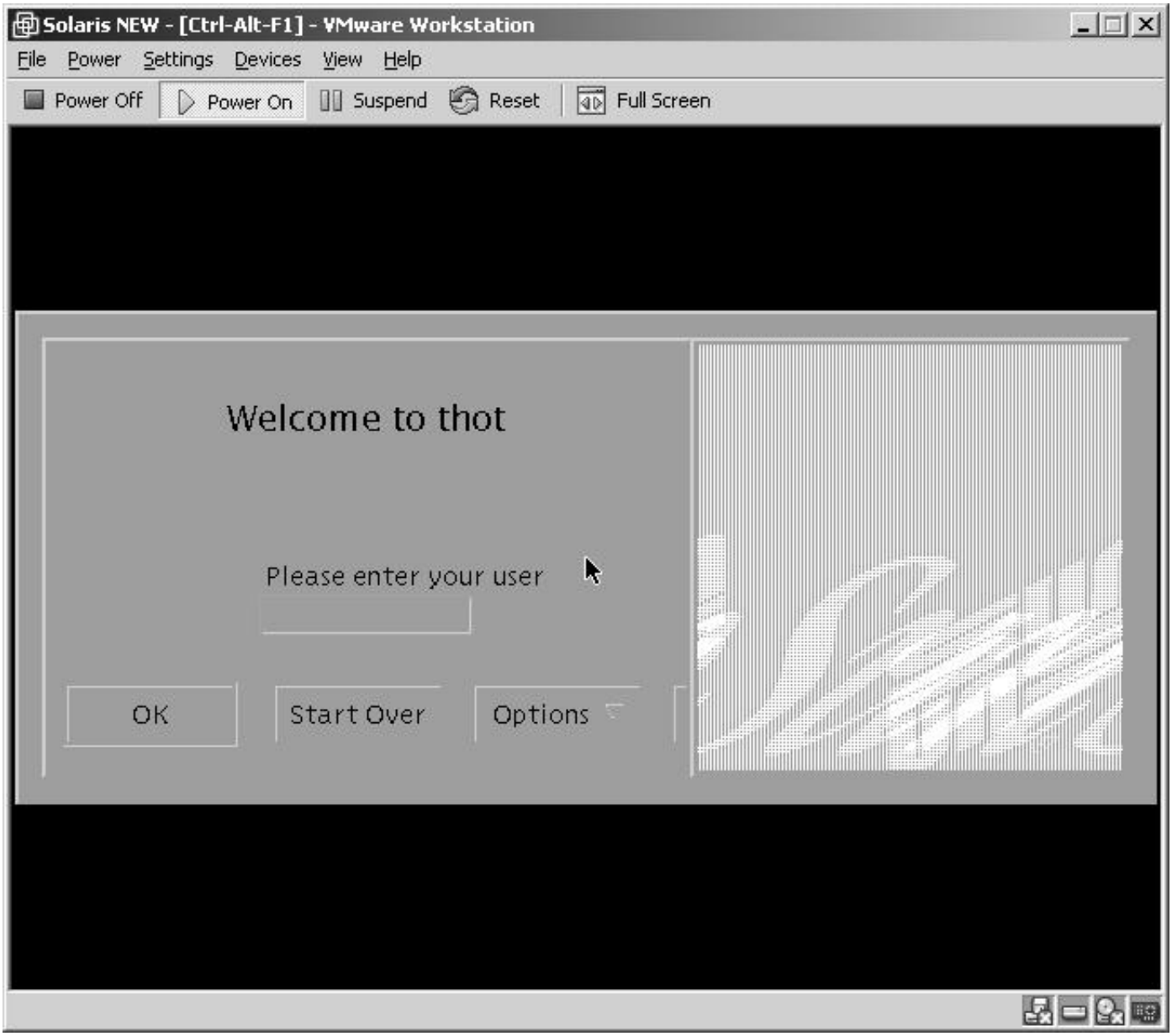

FIG. D.3 - *Ecran de machine virtuelle VMware*

FIG. D.4 - *Ecran de configuration de SEAM*

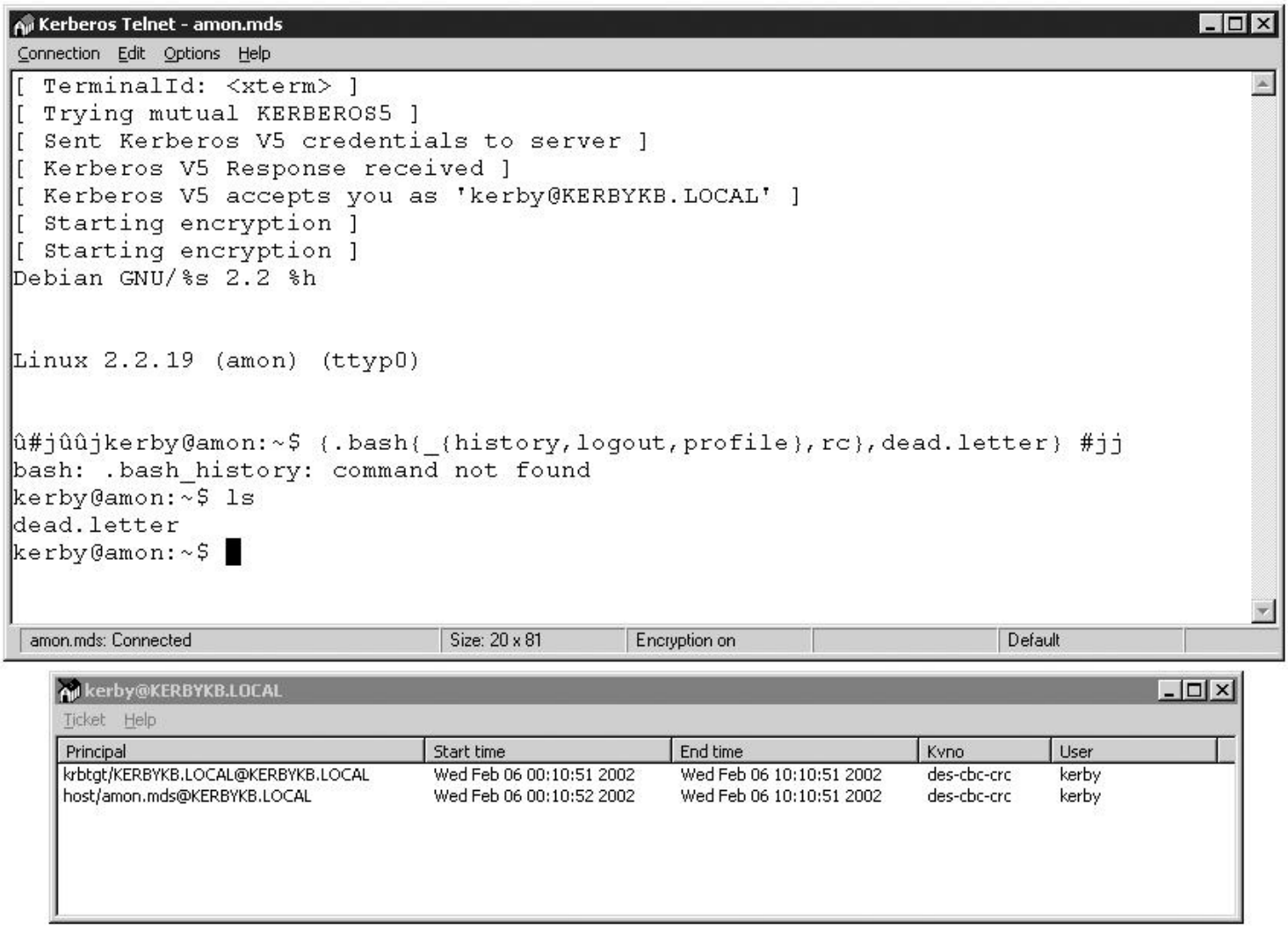

FIG. D.5 - *Exemple d'exécution de KTelnet*

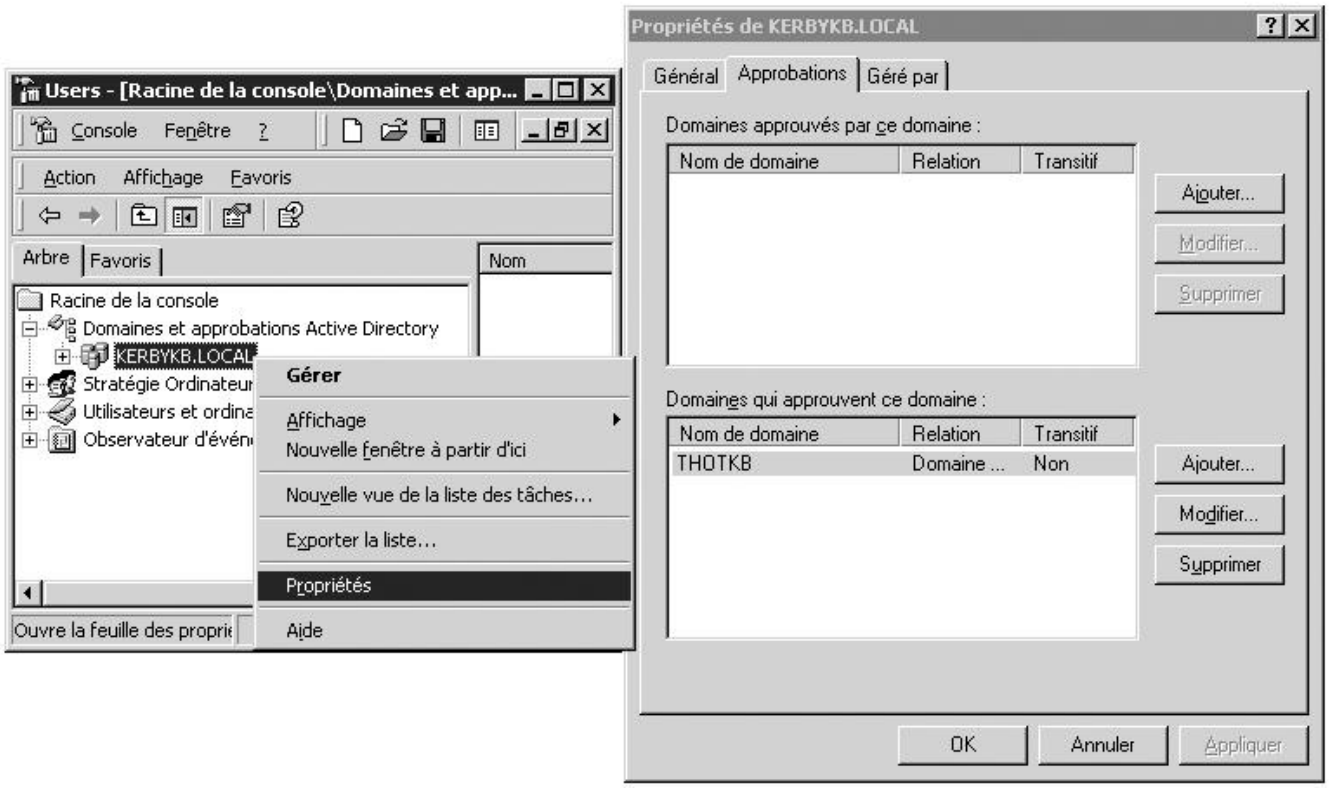

FIG. D.6 - *Configuration d'une confiance unilatérale sous Windows*

Annexe E

Logs de connexions et fichiers de configuration

Cette annexe contient des exemples de fichiers de configuration, et des séquences de connexions telnet, FTP... que nous avons rélisées. Nous y faisons régulièrement référence dans ce document pour illustrer nos propos.

E.1 Fichiers de configuration *krb5.conf*

Les fichiers de configuration suivants sont fréquemment utilisés dans la deuxième partie du document, dans le cas de nos tests. Pour une utilisation plus réelle, nous conseillons la lecture du paragraphe concernant ce fichier, page 51 au chapitre 7. Il aborde le problème du mécanisme de *Single Sign On*.

Exemple 1 (fichier krb5.conf)

```
[libdefaults]
    default_realm = KERBYKB.LOCAL
    default_tkt_enctypes = des-cbc-crc des-cbc-md5
    default_tgs_enctypes = des-cbc-crc des-cbc-md5
    default_etypes_des = des-cbc-crc des-cbc-md5
    default_etypes = des-cbc-crc des-cbc-md5
[realms]
    KERBYKB.LOCAL = {
        kdc = kerby.mds
        admin_server = kerby.mds
        default_domain = mds
    }
[domain_realm]
    mds = KERBYKB.LOCAL
    .mds = KERBYKB.LOCAL
```

Exemple 2 (fichier krb5.conf)

```
[libdefaults]
    default_realm = KERBYKB.LOCAL
    default_tkt_enctypes = des-cbc-crc des-cbc-md5
    default_tgs_enctypes = des-cbc-crc des-cbc-md5
    default_etypes_des = des-cbc-crc des-cbc-md5
    default_etypes = des-cbc-crc des-cbc-md5
[realms]
    KERBYKB.LOCAL = {
         kdc = kerby.mds:88
         default_domain = mds
         admin_server = kerby.mds
    }
[domain_realm]
    .mds = KERBYKB.LOCAL
[appdefaults]
    (inchangé)
```

Exemple 3 (fichier krb5.conf)

```
[libdefaults]
    default_realm = THOTKB
    default_tkt_enctypes = des-cbc-crc des-cbc-md5
    default_tgs_enctypes = des-cbc-crc des-cbc-md5
    default_etypes_des = des-cbc-crc des-cbc-md5
    default_etypes = des-cbc-crc des-cbc-md5
[realms]
    THOTKB = {
kdc = thot.mds
default_domain = mds
admin_server = thot.mds
    }
[domain_realm]
    mds = THOTKB
    .mds = THOTKB
```

E.2 Connexions telnet

Exemple 1 (session de verone à amon.mds)

```
verone:~# ktelnet -l kerby amon.mds
Trying 172.16.8.160...
Connected to amon.mds.
Escape character is '^]'.
[ Trying mutual KERBEROS5 ... ]
[ Kerberos V5 accepts you as ``kerby@KERBYKB.LOCAL'' ]
Debian GNU/%s 2.2 %h
No home directory "/home/kerby"!
Logging in with home = "/".
kerby@amon:/$
```

Exemple 2 (session de thot.mds à amon.mds)

```
root@thot:/# kinit kerby
Password for kerby@KERBYKB.LOCAL:
root@thot:/# ktelnet -l kerby amon.mds
Trying 172.16.8.160...
Connected to amon.mds (172.16.8.160).
Escape character is '^]'.
Segmentation Fault (core dumped)
root@thot:/# klist -5
Ticket cache: /tmp/krb5cc_0
Default principal: kerby@KERBYKB.LOCAL

Valid starting                      Expires                     Service principal
Tue Feb 05 20:28:37 2002  Wed Feb 06 06:28:37 2002  krbtgt/KERBYKB.LOCAL@KERBYKB.LOCAL
        renew until Tue Feb 12 20:28:37 2002
Tue Feb 05 20:28:55 2002  Wed Feb 06 06:28:37 2002  host/amon.mds@KERBYKB.LOCAL
        renew until Tue Feb 12 20:28:37 2002
```

Exemple 3 (session de thot.mds à amon.mds)

```
root@thot:/# kinit kerby
Password for kerby@KERBYKB.LOCAL:
root@thot:/# telnet -l kerby -x -a -f amon.mds
Trying 172.16.8.160...
Connected to amon.mds (172.16.8.160).
Escape character is '^]'.
Waiting for encryption to be negotiated...
[ Kerberos V5 accepts you as ``kerby@KERBYKB.LOCAL'' ]
done.
Debian GNU/%s 2.2 %h
kerby@amon:~$
```

Exemple 4 (session de amon.mds à thot.mds)

```
root@amon:~# kinit kerby
kerby@KERBYKB.LOCAL's Password:
root@amon:~# ktelnet -l kerby thot.mds
Trying 172.16.8.136...
Connected to thot.mds.
Escape character is '^]'.
[ Trying mutual KERBEROS5 ... ]
Connection closed by foreign host.
root@amon:~# klist
Credentials cache: FILE:/tmp/krb5cc_0
        Principal: kerby@KERBYKB.LOCAL

  Issued           Expires          Principal
Feb  5 22:50:57  Feb  6 08:49:36  krbtgt/KERBYKB.LOCAL@KERBYKB.LOCAL
Feb  5 22:51:28  Feb  6 08:49:36  host/thot.mds@KERBYKB.LOCAL
```

Exemple 5 (session de thot.mds à thot.mds)

```
root@thot:/# kinit kerby
Password for kerby@KERBYKB.LOCAL:
root@thot:/# telnet -l kerby thot.mds
Trying 172.16.8.136...
Connected to thot.mds (172.16.8.136).
Escape character is '^]'.
[ Kerberos V5 accepts you as ``kerby@KERBYKB.LOCAL'' ]
Last login: Tue Feb 12 00:33:17 from thot
Sun Microsystems Inc.   SunOS 5.8       Generic February 2000
$
```

Exemple 6 (session de kerby.mds à thot.mds)

```
[ TerminalId: <xterm> ]
[ Trying mutual KERBEROS5 ]
[ Sent Kerberos V5 credentials to server ]
[ Kerberos V5 Response received ]
[ Kerberos V5 accepts you as 'kerby@KERBYKB.LOCAL' ]
[ Starting encryption ]
[ Starting encryption ]
[ Starting encryption ]

    thot (SunOS release 5.8 Generic_108529-11) (pts/3)

IûILast login: Tue Feb 12 12:25:30 from 172.20.0.2
Sun Microsystems Inc.   SunOS 5.8       Generic February 2000
$
```

Exemple 7 (session de KERBYKB à THOTKB)

```
root@thot:/# kinit kerby@KERBYKB.LOCAL
Password for kerby@KERBYKB.LOCAL:
root@thot:/# telnet -l kerby thot.mds
Trying 172.16.8.136...
Connected to thot.mds (172.16.8.136).
Escape character is '^]'.
[ Kerberos V5 accepts you as ``kerby@KERBYKB.LOCAL'' ]
Last login: Wed Feb 13 00:37:03 from 172.20.0.2
Sun Microsystems Inc.   SunOS 5.8       Generic February 2000
$ exit
Connection closed by foreign host.
root@thot:/# klist -5
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: kerby@KERBYKB.LOCAL

Valid starting     Expires            Service principal
02/13/02 07:20:00  02/13/02 17:20:00  krbtgt/KERBYKB.LOCAL@KERBYKB.LOCAL
02/13/02 07:20:00  02/13/02 17:20:00  krbtgt/THOTKB@KERBYKB.LOCAL
02/13/02 07:20:12  02/13/02 17:20:00  host/thot.mds@THOTKB
```

Exemple 8 (session de KERBYKB à THOTKB)

```
root@amon:~# kinit kerby@KERBYKB.LOCAL
kerby@KERBYKB.LOCAL's Password:
root@amon:~# ktelnet -l kerby amon.mds
Trying 172.16.8.160...
Connected to amon.mds.
Escape character is '^]'.
[ Trying mutual KERBEROS5 ... ]
[ Kerberos V5 accepts you as ``kerby@KERBYKB.LOCAL'' ]
Debian GNU/%s 2.2 %h
kerby@amon:~$
```

Exemple 9 (session de amon.mds à thot.mds dans THOTKB)

```
root@amon:/etc# kinit kerby
kerby@THOTKB's Password:
root@amon:/etc# ktelnet -x -l kerby thot.mds
Trying 172.16.8.136...
Connected to thot.mds.
Escape character is '^]'.
[ Trying mutual KERBEROS5 ... ]
[ Kerberos V5 accepts you as ``kerby@THOTKB'' ]
Last login: Wed Feb 13 10:22:49 from amon
Sun Microsystems Inc.   SunOS 5.8       Generic February 2000
$ exit
Connection closed by foreign host.
root@amon:/etc# klist
```

```
Credentials cache: FILE:/tmp/krb5cc_0
        Principal: kerby@THOTKB

  Issued           Expires          Principal
Feb 13 11:22:46  Feb 13 21:22:31  krbtgt/THOTKB@THOTKB
Feb 13 11:22:48  Feb 13 21:22:31  host/thot.mds@THOTKB
```

Exemple 10 (session de thot.mds à amon.mds dans THOTKB)

```
root@thot:/projet/kerby# telnet -x -l kerby amon.mds
Trying 172.16.8.160...
Connected to amon.mds (172.16.8.160).
Escape character is '^]'.
[ Kerberos V5 accepts you as ``kerby@THOTKB'' ]
Debian GNU/%s 2.2 %h

kerby@amon:~$ whoami
kerby
kerby@amon:~$ exit
logout
Connection closed by foreign host.
root@thot:/projet/kerby# klist -5
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: kerby@THOTKB

Valid starting     Expires            Service principal
02/13/02 13:46:45  02/13/02 23:46:45  krbtgt/THOTKB@THOTKB
02/13/02 13:47:08  02/13/02 23:46:45  host/amon.mds@THOTKB
```

E.3 Connexions FTP

Exemple 1 (session de verone à amon.mds)

```
root@verone:~# kftp amon.mds
Connected to amon.mds.
220 amon FTP server (Version 6.00+heimdal-0.21) ready.
Trying GSSAPI...
Authentication successful.

Name (amon.mds:root): kerby
S:232-Linux amon 2.2.19 #1 Sat Jun 9 13:04:06 EST 2001 i686 unknown
S:232-Debian GNU/%s 2.2 %h
S:232 User kerby logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
S:150 Opening ASCII mode data connection for '/bin/ls'.
total 24
drwxr-sr-x    2 kerby    kerby        4096 Jan 31 12:31 .
drwxrwsr-x    5 root     staff        4096 Jan 31 11:38 ..
-rw-------    1 kerby    kerby          71 Feb  1 12:51 .bash_history
-rw-r--r--    1 kerby    kerby         174 Jan 31 11:38 .bash_logout
-rw-r--r--    1 kerby    kerby         373 Jan 31 11:38 .bash_profile
-rw-r--r--    1 kerby    kerby         504 Jan 31 11:38 .bashrc
S:226 Transfer complete.
ftp>
```

Exemple 2 (session de thot.mds à amon.mds)

La ligne "`230 Dumpucko!`" ne se produit que dans le cas spécifique du client SEAM avec le service Heimdal. Elle est expliquée en 3.2.2 page 32.

```
root@thot:/# kftp amon.mds
Connected to amon.mds.
220 amon FTP server (Version 6.00+heimdal-0.21) ready.
334 Send authorization data.
GSSAPI accepted as authentication type
GSSAPI authentication succeeded
Using kerberos_v5 mechanism type
Name (amon.mds:root): kerby
232-Linux amon 2.2.19 #1 Sat Jun 9 13:04:06 EST 2001 i686 unknown
232-Debian GNU/%s 2.2 %h
232 User kerby logged in.
230 Dumpucko!
Remote system type is UNIX.
Using ascii mode to transfer files.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 0
```

```
226 Transfer complete.
ftp>
```

Exemple 3 (session de thot.mds à amon.mds)

```
root@thot:/# ftp -x amon.mds
Connected to amon.mds.
220 amon FTP server (Version 6.00+heimdal-0.21) ready.
334 Send authorization data.
GSSAPI accepted as authentication type
GSSAPI authentication succeeded
200 Data protection is private.
Name (amon.mds:root): kerby
232-Linux amon 2.2.19 #1 Sat Jun 9 13:04:06 EST 2001 i686 unknown
232-Debian GNU/%s 2.2 %h
232 User kerby logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
total 32
drwxr-sr-x    2 kerby    kerby        4096 Feb  6 00:24 .
drwxrwsr-x    5 root     staff        4096 Jan 31 11:38 ..
-rw-------    1 kerby    kerby        4135 Feb 10 18:24 .bash_history
-rw-r--r--    1 kerby    kerby         174 Jan 31 11:38 .bash_logout
-rw-r--r--    1 kerby    kerby         373 Jan 31 11:38 .bash_profile
-rw-r--r--    1 kerby    kerby         504 Jan 31 11:38 .bashrc
-rw-------    1 kerby    kerby           3 Feb  6 00:24 dead.letter
226 Transfer complete.
ftp>
```

Exemple 4 (session de kerby.mds à amon.mds)

```
[ FTP Connected to amon.mds ]
<-- 220 amon FTP server (Version 6.00+heimdal-0.21) ready.
--> AUTH GSSAPI
<-- 334 Send authorization data.
--> ADAT <....>
<-- 235 ADAT=<....>
--> USER kerby
<-- S:232-Linux amon 2.2.19 #1 Sat Jun 9 13:04:06 EST 2001 i686 unknown
<-- S:232-
<-- S:232-Most of the programs included with the Debian GNU/Linux system are
<-- S:232-freely redistributable; the exact distribution terms for each program
<-- S:232-are described in the individual files in /usr/doc/*/copyright
<-- S:232-
<-- S:232-Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
<-- S:232-permitted by applicable law.
<-- S:232-Debian GNU/%s 2.2 %h
```

```
<-- S:232 User kerby logged in.
FTP>ls
--> TYPE A
<-- S:200 Type set to A.
--> PORT 172,16,8,209,5,229
<-- S:200 PORT command successful.
--> LIST
<-- S:150 Opening ASCII mode data connection for '/bin/ls'.
total 32
drwxr-sr-x    2 kerby    kerby        4096 Feb  6 00:24 .
drwxrwsr-x    5 root     staff        4096 Jan 31 11:38 ..
-rw-------    1 kerby    kerby        4102 Feb  8 00:28 .bash_history
-rw-r--r--    1 kerby    kerby         174 Jan 31 11:38 .bash_logout
-rw-r--r--    1 kerby    kerby         373 Jan 31 11:38 .bash_profile
-rw-r--r--    1 kerby    kerby         504 Jan 31 11:38 .bashrc
-rw-------    1 kerby    kerby           3 Feb  6 00:24 dead.letter

Transfer complete, 475 bytes
<-- S:226 Transfer complete.
FTP>
```

Exemple 5 (session de thot.mds à thot.mds)

```
root@thot:/# kftp thot.mds
Connected to thot.mds.
220 thot FTP server (SunOS 5.8) ready.
334 Using authentication type GSSAPI; ADAT must follow
GSSAPI accepted as authentication type
GSSAPI authentication succeeded
Using kerberos_v5 mechanism type
Name (thot.mds:root): kerby
232 GSSAPI user kerby@KERBYKB.LOCAL is authorized as kerby
230 User kerby logged in.
Remote system type is UNIX.
Using ascii mode to transfer files.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 0
226 Transfer complete.
ftp>
```

Exemple 6 (session de amon.mds à thot.mds)

```
root@amon:~# kftp thot.mds
Connected to thot.mds.
220 thot FTP server (SunOS 5.8) ready.
Trying GSSAPI...
Authentication successful.
```

```
Name (thot.mds:root): kerby
S:232 GSSAPI user kerby@KERBYKB.LOCAL is authorized as kerby
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
S:200 PORT command successful.
S:530 Please login with USER and PASS.
ftp>
```

Exemple 7 (session de KERBYKB à THOTKB)

```
root@thot:/# ftp -x thot.mds
Connected to thot.mds.
220 thot FTP server (Version 5.60) ready.
334 Using authentication type GSSAPI; ADAT must follow
GSSAPI accepted as authentication type
GSSAPI authentication succeeded
200 Data channel protection level set to private.
Name (thot.mds:root): kerby
232 GSSAPI user kerby@KERBYKB.LOCAL is authorized as kerby
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 4
-rw-------   1 other        253 fév 13 07:20 .bash_history
-rw-r--r--   1 other         20 fév 13 00:11 .k5login
226 Transfer complete.
```

Exemple 8 (session de amon.mds à thot.mds dans THOTKB)

```
root@amon:/etc# kftp thot.mds
Connected to thot.mds.
220 thot FTP server (Version 5.60) ready.
Trying GSSAPI...
Authentication successful.

Name (thot.mds:root): kerby
S:232 GSSAPI user kerby@THOTKB is authorized as kerby
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
S:200 PORT command successful.
S:150 Opening ASCII mode data connection for /bin/ls.
total 4
-rw-------   1 other        356 fév 13 10:16 .bash_history
-rw-r--r--   1 other         33 fév 13 10:21 .k5login
S:226 Transfer complete.
```

```
ftp> quit
S:221 Goodbye.
root@amon:/etc# klist
Credentials cache: FILE:/tmp/krb5cc_0
        Principal: kerby@THOTKB

  Issued           Expires          Principal
Feb 13 11:22:46  Feb 13 21:22:31  krbtgt/THOTKB@THOTKB
Feb 13 11:22:48  Feb 13 21:22:31  host/thot.mds@THOTKB
Feb 13 11:34:57  Feb 13 21:22:31  ftp/thot.mds@THOTKB
```

Annexe F

Traces réseau

Nous listons ici le résultat d'une exécution de *tcpdump* pour illustrer concrètement les échanges qui ont lieu entre les différents intervenants du protocole. Nous sommes dans la configuration où le KDC est un serveur Windows 2000 (de nom *KERBY*), le client est sur le système Solaris *thot.mds*, et il établit une connexion telnet cryptée avec le serveur *amon.mds*. Le nom d'utilisateur est *kerby@KERBYKB.LOCAL*.

F.1 Obtention d'un TGT

Nous exécutons ici la commande *kinit* sur *thot.mds*. Avant de pouvoir saisir un mot de passe, ce programme client s'adresse au KDC pour vérifier, entre autres, que les horloges sont bien synchronisées. Le datagramme UDP ci-dessous est donc immédiatement envoyé à *KERBY*. Notons qu'il y figure bien le nom d'utilisateur (`kerby`), le domaine (`KERBYKB.LOCAL`), le ticket demandé (un TGT correspond au service `krbtgt`), la date du client (`20020218204326` pour le 18 février 2002 à 20:43:26) et la date de péremption maximale proposée (8 heures plus tard).

```
20:43:26.179433 thot.mds.32945 > KERBY.kerberos:  v5 (DF)
0x0000  4500 00cf 30cf 4000 ff11 e0d4 ac10 0888 E...0.@.........
0x0010  ac10 08d1 80b1 0058 00bb 2745 6a81 b030 .......X..'Ej..0
0x0020  81ad a103 0201 05a2 0302 010a a481 a030 ...............0
0x0030  819d a007 0305 0000 0000 00a1 1230 10a0 ............0..
0x0040  0302 0101 a109 3007 1b05 6b65 7262 79a2 ......0...kerby.
0x0050  0f1b 0d4b 4552 4259 4b42 2e4c 4f43 414c ...KERBYKB.LOCAL
0x0060  a322 3020 a003 0201 00a1 1930 171b 066b ."0........0...k
0x0070  7262 7467 741b 0d4b 4552 4259 4b42 2e4c rbtgt..KERBYKB.L
0x0080  4f43 414c a411 180f 3230 3032 3032 3138 OCAL....20020218
0x0090  3230 3433 3236 5aa5 1118 0f32 3030 3230 204326Z....20020
0x00a0  3231 3930 3634 3332 365a a706 0204 3c71 219064326Z....<q
0x00b0  676e a808 3006 0201 0102 0103 a911 300f gn..0.........0.
0x00c0  300d a003 0201 02a1 0604 04ac 1008 88   0..............
```

Le KDC répond alors, en indiquant sa propre heure. Le TGT n'apparaît pas encore ici.

```
20:43:26.187003 KERBY.kerberos > thot.mds.32945:
0x0000  4500 00e3 d8c6 0000 8011 f7c9 ac10 08d1 E...............
0x0010  ac10 0888 0058 80b1 00cf 6fa7 7e81 c430 .....X....o.~..0
0x0020  81c1 a003 0201 05a1 0302 011e a411 180f ................
0x0030  3230 3032 3032 3138 3230 3435 3238 5aa5 200202182045 28Z.
0x0040  0502 030f 07e4 a603 0201 19a9 0f1b 0d4b ...............K
0x0050  4552 4259 4b42 2e4c 4f43 414c aa22 3020 ERBYKB.LOCAL."0.
0x0060  a003 0201 02a1 1930 171b 066b 7262 7467 .......0...krbtg
0x0070  741b 0d4b 4552 4259 4b42 2e4c 4f43 414c t..KERBYKB.LOCAL
0x0080  ac61 045f 305d 3045 a103 0201 0ba2 3e04 .a._0]0E......>.
0x0090  3c30 3a30 1ba0 0302 0103 a114 0412 4b45 <0:0..........KE
0x00a0  5242 594b 422e 4c4f 4341 4c6b 6572 6279 RBYKB.LOCALkerby
0x00b0  301b a003 0201 01a1 1404 124b 4552 4259 0..........KERBY
0x00c0  4b42 2e4c 4f43 414c 6b65 7262 7930 09a1 KB.LOCALkerby0..
0x00d0  0302 0102 a202 0400 3009 a103 0201 0fa2 ........0.......
0x00e0  0204 00                                 ...
```

L'utilisateur tape alors son mot de passe avec *kinit* ; celui-ci envoie le datagramme suivant au KDC. On y retrouve des informations similaires au premier paquet. Rappelons que le mot de passe ne figure pas ici, même pas crypté.

```
20:43:28.157199 thot.mds.32946 > KERBY.kerberos:  v5 (DF)
0x0000  4500 0122 30d0 4000 ff11 e080 ac10 0888 E.."0.@.........
0x0010  ac10 08d1 80b2 0058 010e e8b6 6a82 0102 .......X....j...
0x0020  3081 ffa1 0302 0105 a203 0201 0aa3 5030 0.............P0
0x0030  4e30 4ca1 0302 0102 a245 0443 3041 a003 N0L......E.C0A..
0x0040  0201 03a2 3a04 387e 5fae 1374 46f1 92b4 ....:.8~_..tF...
0x0050  5791 10af 1339 9700 aa94 03c9 66ad 41de W....9......f.A.
0x0060  030e 3455 04dc 4ef5 8c41 4d43 0f32 7c74 ..4U..N..AMC.2|t
0x0070  bd8d 08b2 2c41 ccbd f9e6 d53d 23b5 06a4 ....,A.....=#...
0x0080  81a0 3081 9da0 0703 0500 0000 0000 a112 ..0.............
0x0090  3010 a003 0201 01a1 0930 071b 056b 6572 0........0...ker
0x00a0  6279 a20f 1b0d 4b45 5242 594b 422e 4c4f by....KERBYKB.LO
0x00b0  4341 4ca3 2230 20a0 0302 0100 a119 3017 CAL."0........0.
0x00c0  1b06 6b72 6274 6774 1b0d 4b45 5242 594b ..krbtgt..KERBYK
0x00d0  422e 4c4f 4341 4ca4 1118 0f32 3030 3230 B.LOCAL....20020
0x00e0  3231 3832 3034 3332 365a a511 180f 3230 218204326Z....20
0x00f0  3032 3032 3139 3036 3433 3236 5aa7 0602 020219064326Z...
0x0100  043c 7167 70a8 0830 0602 0101 0201 03a9 .<qgp..0........
0x0110  1130 0f30 0da0 0302 0102 a106 0404 ac10 .0.0............
0x0120  0888                                    ..
```

Le serveur envoie enfin le TGT, toujours en utilisant le protocole UDP. Nous constatons qu'il reprend à nouveau des informations comme le nom du *principal* et le domaine, mais les autres informations sont cryptées (comme la date de péremption). Pour des raisons de lisibilité, nous avons retiré 658 octets de la trace de ce paquet ; nous l'analyserons à nouveau lors de la demande de ticket de service.

```
20:43:28.170947 KERBY.kerberos > thot.mds.32946:  v5
0x0000  4500 04ff d8c7 0000 8011 f3ac ac10 08d1 E...............
0x0010  ac10 0888 0058 80b2 04eb 8ad9 6b82 04df .....X......k...
0x0020  3082 04db a003 0201 05a1 0302 010b a21f 0...............
0x0030  301d 301b a103 0201 03a2 1404 124b 4552 0.0.........KER
0x0040  4259 4b42 2e4c 4f43 414c 6b65 7262 79a3 BYKB.LOCALkerby.
0x0050  0f1b 0d4b 4552 4259 4b42 2e4c 4f43 414c ...KERBYKB.LOCAL
0x0060  a412 3010 a003 0201 01a1 0930 071b 056b ..0........0...k
0x0070  6572 6279 a582 037f 6182 037b 3082 0377 erby....a..{0..w
0x0080  a003 0201 05a1 0f1b 0d4b 4552 4259 4b42 .........KERBYKB
0x0090  2e4c 4f43 414c a222 3020 a003 0201 00a1 .LOCAL."0.......
0x00a0  1930 171b 066b 7262 7467 741b 0d4b 4552 .0...krbtgt..KER
0x00b0  4259 4b42 2e4c 4f43 414c a382 0339 3082 BYKB.LOCAL...90.
0x00c0  0335 a003 0201 01a2 8203 2c04 8203 2837 .5........,...(7
0x00d0  1a5a e46e 6058 4b61 2e42 db46 7118 f895 .Z.n`XKa.B.Fq...
0x00e0  8c14 98bc e21a 035b 4199 3124 4e53 e214 .......[A.1$NS..
0x00f0  ae52 5bca b95e 2b47 37c0 f523 e631 89d4 .R[..^+G7..#.1..
0x0100  e669 19fa 01ea e5a4 ce16 8437 7375 f051 .i.........7su.Q
                          ...
0x03a0        15ef 0cbd d86b 63a2 2eb1 a301 57e6   .....kc.....W.
0x03b0  f9b4 4bfa c208 dcd7 49e5 a8b7 5acb cf48 ..K.....I...Z..H
0x03c0  dd9b 8b3b 793c 606e cc97 d9ea 25d8 3769 ...;y<`n....%.7i
0x03d0  42d3 bb66 54f0 a3a0 d614 b878 20a1 9a83 B..fT......x....
0x03e0  504b 4a1d 72c3 cf29 56e1 f063 baee 5991 PKJ.r..)V..c..Y.
0x03f0  92c0 eacd 25d9 6ca6 8201 0430 8201 00a0 ....%.l....0....
0x0400  0302 0101 a103 0201 01a2 81f3 0481 f0c4 ................
0x0410  d9cd 9f93 28ee ee06 c3e1 51c3 a50f 7a64 ....(.....Q...zd
0x0420  4a2a 6720 1fcb 9a4b c3fc 3d88 ae8f 01b6 J*g....K..=.....
0x0430  a0f4 4c14 dfa1 3fe5 e4b2 bb09 864e 8cf3 ..L...?......N..
0x0440  2e10 3e9c aa56 3fa7 4cb8 b540 06a1 7374 ..>..V?.L..@..st
0x0450  fd40 e2ea 5954 fd9a 5cbf caec fdee 326e .@..YT..\.....2n
0x0460  bc36 33d6 abc0 10c9 833c e72d 31cd a5ee .63......<.-1...
0x0470  7c15 1d2f 8200 490b 7e69 6d87 4322 7eea |../..I.~im.C"~.
0x0480  70ae 83b7 6596 93cc 1859 34dd 972f 47aa p...e....Y4../G.
0x0490  0bd6 0f09 c4fd b72d cc99 553e 77dd 0f5c .......-..U>w..\
0x04a0  eced aa4a 9543 1c06 a1c6 f7e9 908f 6792 ...J.C........g.
0x04b0  c246 bda9 ccc8 3b8e eb85 ba44 b011 d970 .F....;....D...p
0x04c0  8310 13ea 9584 7417 a891 750c 9c1b cdca ......t...u.....
0x04d0  1235 5287 27a2 998e 8616 a699 9aa8 1f0e .5R.'...........
0x04e0  68ac da53 d06d a595 25e8 b337 1967 6dcd h..S.m..%..7.gm.
0x04f0  8987 5fe7 1890 bf3c 00c4 bf89 1921 d0   .._....<.....!.
```

F.2 Demande de ticket pour le telnet

A présent l'utilisateur souhaite se connecter au service telnet de *amon.mds*; il utilise (avec le telnet du MIT) la commande

```
telnet -x -l kerby -k KERBYKB.LOCAL amon.mds
```

Le client envoie alors un paquet UDP pour demander le ticket correspondant. On y reconnaît le TGT obtenu précédemment (nous avons retiré ci-dessous

les même 658 octets, il est possible de reconnaître les octets identiques autour de la coupure ci-dessous et dans le paquet UDP de la page 91), et le nom du ticket demandé (on peut observe les mots de la chaîne de caractères *host/amon.mds@KERBYKB.LOCAL*).

```
20:43:38.060467 thot.mds.32947 > KERBY.kerberos:  (DF)
0x0000  4500 04dc 30d1 4000 ff11 dcc5 ac10 0888 E...0.@.........
0x0010  ac10 08d1 80b3 0058 04c8 0d6c 6c82 04bc .......X...ll...
0x0020  3082 04b8 a103 0201 05a2 0302 010c a382 0...............
0x0030  043a 3082 0436 3082 0432 a103 0201 01a2 .:0..60..2......
0x0040  8204 2904 8204 256e 8204 2130 8204 1da0 ..)...%n..!0....
0x0050  0302 0105 a103 0201 0ea2 0703 0500 0000 ................
0x0060  0000 a382 037f 6182 037b 3082 0377 a003 ......a..{0..w..
0x0070  0201 05a1 0f1b 0d4b 4552 4259 4b42 2e4c .......KERBYKB.L
0x0080  4f43 414c a222 3020 a003 0201 00a1 1930 OCAL."0........0
0x0090  171b 066b 7262 7467 741b 0d4b 4552 4259 ...krbtgt..KERBY
0x00a0  4b42 2e4c 4f43 414c a382 0339 3082 0335 KB.LOCAL...90..5
0x00b0  a003 0201 01a2 8203 2c04 8203 2837 1a5a ........,...(7.Z
0x00c0  e46e 6058 4b61 2e42 db46 7118 f895 8c14 .n`XKa.B.Fq.....
0x00d0  98bc e21a 035b 4199 3124 4e53 e214 ae52 .....[A.1$NS...R
0x00e0  5bca b95e 2b47 37c0 f523 e631 89d4 e669 [..^+G7..#.1...i
0x00f0  19fa 01ea e5a4 ce16 8437 7375 f051      .........7su.Q
                            ...
0x0390  15ef 0cbd d86b 63a2 2eb1 a301 57e6 f9b4 .....kc.....W...
0x03a0  4bfa c208 dcd7 49e5 a8b7 5acb cf48 dd9b K.....I...Z..H..
0x03b0  8b3b 793c 606e cc97 d9ea 25d8 3769 42d3 .;y<`n....%.7iB.
0x03c0  bb66 54f0 a3a0 d614 b878 20a1 9a83 504b .fT......x....PK
0x03d0  4a1d 72c3 cf29 56e1 f063 baee 5991 92c0 J.r..)V..c..Y...
0x03e0  eacd 25d9 6ca4 8184 3081 81a0 0302 0101 ..%.l...0.......
0x03f0  a27a 0478 035e 69eb 7f52 19ca 559f e806 .z.x.^i..R..U...
0x0400  f761 4078 b969 be8f b143 83d4 5f1e 9e25 .a@x.i...C.._..%
0x0410  52bb ae92 34f0 b8ba 6bbd 100e ee3d c21c R...4...k....=..
0x0420  361a ada9 f129 a723 078a 9bd4 152c 4892 6....).#.....,H.
0x0430  6900 345f f936 d009 53bf 8db1 f619 e479 i.4_.6..S......y
0x0440  3e61 5adc f65e 3985 8507 2604 a23f 1681 >aZ..^9...&..?..
0x0450  0805 4686 6795 d368 f6ae 84b7 d861 2369 ..F.g..h.....a#i
0x0460  4370 560e 3e2c e52c 78a2 2b71 a46e 306c CpV.>,.,x.+q.n0l
0x0470  a007 0305 0000 0000 00a2 0f1b 0d4b 4552 .............KER
0x0480  4259 4b42 2e4c 4f43 414c a31b 3019 a003 BYKB.LOCAL..0...
0x0490  0201 03a1 1230 101b 0468 6f73 741b 0861 .....0...host..a
0x04a0  6d6f 6e2e 6d64 73a5 1118 0f32 3030 3230 mon.mds....20020
0x04b0  3231 3930 3634 3332 365a a706 0204 3c71 219064326Z....<q
0x04c0  677a a805 3003 0201 01a9 1130 0f30 0da0 gz..0......0.0..
0x04d0  0302 0102 a106 0404 ac10 0888           ............
```

La réponse du serveur contient naturellement le ticket de service demandé. Celui-ci a été raccourci (de 656 octets) pour des raisons de lisibilité. On reconnaît également le nom du *principal*: *host/amon.mds*.

```
20:43:38.070367 KERBY.kerberos > thot.mds.32947:
0x0000  4500 04a5 d8c8 0000 8011 f405 ac10 08d1 E...............
0x0010  ac10 0888 0058 80b3 0491 ba57 6d82 0485 .....X.....Wm...
0x0020  3082 0481 a003 0201 05a1 0302 010d a30f 0...............
0x0030  1b0d 4b45 5242 594b 422e 4c4f 4341 4ca4 ..KERBYKB.LOCAL.
0x0040  1230 10a0 0302 0101 a109 3007 1b05 6b65 .0........0...ke
0x0050  7262 79a5 8203 7861 8203 7430 8203 70a0 rby...xa..t0..p.
0x0060  0302 0105 a10f 1b0d 4b45 5242 594b 422e ........KERBYKB.
0x0070  4c4f 4341 4ca2 1b30 19a0 0302 0103 a112 LOCAL..0........
0x0080  3010 1b04 686f 7374 1b08 616d 6f6e 2e6d 0...host..amon.m
0x0090  6473 a382 0339 3082 0335 a003 0201 01a2 ds...90..5......
0x00a0  8203 2c04 8203 2859 ee04 0880 4dc9 30ce ..,...(Y....M.0.
0x00b0  c313 b35f 971e af6a 50b7 6a18 b655 71de ..._...jP.j..Uq.
0x00c0  f8e4 39c2 b446 6886 195e 08e7 0656 ea24 ..9..Fh..^...V.$
0x00d0  ff5f 7fb5 2d7c 0c79 990e 41d9 9332 c46f ._..-|.y..A..2.o
                          ...
0x0370  0b90 f1ba ebba aaaa 6825 21e8 2def 7f94 ........h%!.-...
0x0380  3c48 6216 da42 bef8 c102 fc42 dc35 3a94 <Hb..B.....B.5:.
0x0390  0c09 5fa3 21e0 18fd a80d 655e 78b6 957e .._.!.....e^x..~
0x03a0  4a22 0b84 c018 223a bc67 9652 c467 6f47 J".....":.g.R.goG
0x03b0  a6e8 751c 8f88 61c9 3895 2b6a ed12 9bad ..u...a.8.+j....
0x03c0  1d98 2e27 337b 8e54 075c fef5 6f17 27a6 ...'3{.T.\..o.'.
0x03d0  81d3 3081 d0a0 0302 0101 a103 0201 01a2 ..0.............
0x03e0  81c3 0481 c07f c8b2 63e6 b8f0 7cd3 2370 ........c...|.#p
0x03f0  b40e e8b9 7698 5006 461a 8865 c7d4 ffd4 ....v.P.F..e....
0x0400  eca4 55bc 66e1 1448 f9c5 c237 f0d3 f0cb ..U.f..H...7....
0x0410  d496 1d92 56f1 31ac 534a 54e9 8d30 e0c5 ....V.1.SJT..0..
0x0420  ec6a 1a00 1944 f999 066d 2d82 f249 7f33 .j...D...m-..I.3
0x0430  4f14 4e21 88ea 3975 04f9 ec48 ce2a 63db O.N!..9u...H.*c.
0x0440  aefa 6142 66ca a5ca 5779 0c7f d965 4d83 ..aBf...Wy...eM.
0x0450  5150 f828 558c 5079 7f5d f0db 4260 4b98 QP.(U.Py.]..B`K.
0x0460  248b 69de 761c c2e3 c9c9 015c 491f 9227 $.i.v......\I..'
0x0470  6c96 f39c ebf8 62c8 a96b 8c6d 0999 d999 l.....b..k.m....
0x0480  bb17 9281 9c3b ef74 df52 b87e d9f3 81f6 .....;.t.R.~....
0x0490  1183 97d6 e845 090c 167a 0402 68bc d9da .....E...z..h...
0x04a0  75f1 1591 c6                            u....
```

F.3 Authentification auprès du service

A présent, le client telnet peut s'adresser au serveur et s'y authentifier à l'aide du ticket. Les traces suivantes sont donc maintenant en TCP. Nous coupons ici le début du protocole (inexploitable à la lecture) où se négocie le service telnet. Le premier paquet listé ci-dessous est émis par le client, et contient le ticket acquis ci-dessus (page 93). Nous l'avons raccourci de 657 octets, pour que la coupure corresponde à celle opérée précedemment (notons que le ticket a été augmenté de 1 octet : en effet, un `0xFF` figure dans ce ticket alors qu'il n'apparaissait pas dans celui émis par le KDC).

```
20:43:38.075139 thot.mds.32800 > amon.mds.telnet: P 37:1155(1118) ack 55 win 33304
<nop,nop,timestamp 4810443 24813359> (DF) [tos 0x10]
0x0000  4510 0492 b162 4000 4006 1bab ac10 0888 E....b@.@.......
0x0010  ac10 08a0 8020 0017 deb1 b8db 645a c149 ............dZ.I
0x0020  8018 8218 8ff3 0000 0101 080a 0049 66cb .............If.
0x0030  017a 9f2f fffa 2503 6b65 7262 79ff f0ff .z./..%.kerby...
0x0040  fa25 0002 0200 6e82 042c 3082 0428 a003 .%....n..,0..(..
0x0050  0201 05a1 0302 010e a207 0305 0020 0000 ................
0x0060  00a3 8203 7861 8203 7430 8203 70a0 0302 ....xa..t0..p...
0x0070  0105 a10f 1b0d 4b45 5242 594b 422e 4c4f ......KERBYKB.LO
0x0080  4341 4ca2 1b30 19a0 0302 0103 a112 3010 CAL..0........0.
0x0090  1b04 686f 7374 1b08 616d 6f6e 2e6d 6473 ..host..amon.mds
0x00a0  a382 0339 3082 0335 a003 0201 01a2 8203 ...90..5........
0x00b0  2c04 8203 2859 ee04 0880 4dc9 30ce c313 ,...(Y....M.0...
0x00c0  b35f 971e af6a 50b7 6a18 b655 71de f8e4 ._...jP.j..Uq...
0x00d0  39c2 b446 6886 195e 08e7 0656 ea24 ffff 9..Fh..^...V.$..
0x00e0  5f7f b52d 7c0c 7999 0e41 d993 32c4 6f   _..-|.y..A..2.o
                          ...
0x0380  0b90 f1ba ebba aaaa 6825 21e8 2def 7f94 ........h%!.-...
0x0390  3c48 6216 da42 bef8 c102 fc42 dc35 3a94 <Hb..B.....B.5:.
0x03a0  0c09 5fa3 21e0 18fd a80d 655e 78b6 957e ..._.!.....e^x..~
0x03b0  4a22 0b84 c018 223a bc67 9652 c467 6f47 J".....":.g.R.goG
0x03c0  a6e8 751c 8f88 61c9 3895 2b6a ed12 9bad ..u...a.8.+j....
0x03d0  1d98 2e27 337b 8e54 075c fef5 6f17 27a4 ...'3{.T.\..o.'.
0x03e0  8196 3081 93a0 0302 0101 a281 8b04 8188 ..0.............
0x03f0  b87d 5543 728e 95c0 1b0c 0ca4 d7a5 8fcd .}UCr...........
0x0400  9ddb 7e5c 8213 2c60 2bd8 0831 6a02 60de ..~\..,`+..1j.`.
0x0410  f82c df0b b19f dc20 0140 8351 acdd 2af2 .,....... .@.Q..*.
0x0420  7ca1 f516 252f 2b82 a045 73f1 17f1 250d |...%/+..Es...%.
0x0430  d5d4 ceea ed40 8be5 5bb9 4f3d 89b6 dd48 .....@..[.O=...H
0x0440  c4ea 170d ffff 506e e08b 3030 04f8 ac72 ......Pn..00...r
0x0450  470d 69e5 b30b cc59 52ad 405a b9f6 6fc8 G.i....YR.@Z..o.
0x0460  374d fb3a eaf3 7994 acdf 106d 532a fba1 7M.:..y....mS*..
0x0470  809a d598 70e7 6bd1 31ff f0ff fa26 05ff ....p.k.1....&..
0x0480  f0ff fa26 0101 02ff f0ff fa1f 0050 0018 ...&.........P..
0x0490  fff0                                    ..
20:43:38.146800 amon.mds.telnet > thot.mds.32800: . ack 1155 win 16060
<nop,nop,timestamp 24813369 4810443> (DF)
0x0000  4500 0034 53a3 4000 4006 7dd8 ac10 08a0 E..4S.@.@.}.....
0x0010  ac10 0888 0017 8020 645a c149 deb1 bd39 ........dZ.I...9
0x0020  8010 3ebc 8529 0000 0101 080a 017a 9f39 ..>..).......z.9
0x0030  0049 66cb                               .If.
20:43:43.105241 amon.mds.telnet > thot.mds.32800: P 55:183(128) ack 1155 win 16060
<nop,nop,timestamp 24813865 4810443> (DF)
0x0000  4500 00b4 53a8 4000 4006 7d53 ac10 08a0 E...S.@.@.}S....
0x0010  ac10 0888 0017 8020 645a c149 deb1 bd39 ........dZ.I...9
0x0020  8018 3ebc 1be4 0000 0101 080a 017a a129 ..>..........z.)
0x0030  0049 66cb fffa 2502 0202 036f 4930 47a0 .If...%....oI0G.
0x0040  0302 0105 a103 0201 0fa2 3b30 39a0 0302 ..........;09...
0x0050  0101 a232 0430 05f2 2b5c 054e 8b36 006a ...2.0..+\.N.6.j
```

```
0x0060  180a 150e 2e2b 8bd8 69fb 5cdd 6496 1822 .....+..i.\.d.."
0x0070  1b77 efe7 daa3 a07a a9e8 8b3a 82bc f883 .w.....z...:....
0x0080  7ee5 99c7 0f65 fff0 fffa 2502 0202 026b ~....e....%....k
0x0090  6572 6279 404b 4552 4259 4b42 2e4c 4f43 erby@KERBYKB.LOC
0x00a0  414c fff0 fffa 2600 0101 80fa b475 e8c5 AL....&......u..
0x00b0  815d fff0                               .]..
```

La suite des paquets TCP est illisible : la communication telnet est à présent cryptée.

Annexe G

Erreur d'exécution

Comme nous l'avons vu au chapitre 3 page 29, SEAM pose de graves problèmes d'interopérabilité : les connexions telnet se romptent avec des *segmentation faults*. Cette annexe propose ici l'exécution du *backtrace* de *gdb* pour éventuellement comprendre où les erreurs se produisent.

Nous avons testé le client SEAM disponible, avec le service telnet de Heimdal[1] : il s'agit de */usr/krb5/bin/telnet*. Voici l'analyse du *core* généré lors de l'exécution de ce client :

```
root@thot:/# gdb --se=/usr/krb5/bin/telnet --core=core
GNU gdb 5.0
Copyright 2000 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i386-pc-solaris2.8"...(no debugging symbols found)...
Core was generated by `/usr/krb5/bin/telnet -l kerby -k KERBYKB.LOCAL amon.mds'.
Program terminated with signal 11, Segmentation Fault.
Reading symbols from /usr/lib/gss/do/mech_krb5.so.1...(no debugging symbols found)...done.
Loaded symbols for /usr/lib/gss/do/mech_krb5.so.1
Reading symbols from /lib/libnsl.so.1...(no debugging symbols found)...done.
...
Reading symbols from /lib/nss_files.so.1...(no debugging symbols found)...done.
Loaded symbols for /lib/nss_files.so.1
#0  0xdfb78e4a in krb5_rd_rep () from /usr/lib/gss/do/mech_krb5.so.1
(gdb) bt
#0  0xdfb78e4a in krb5_rd_rep () from /usr/lib/gss/do/mech_krb5.so.1
#1  0x805b835 in kerberos5_reply ()
#2  0x8058e0b in auth_reply ()
#3  0x805457d in suboption ()
#4  0x8055bee in telrcv ()
#5  0x80560dd in Scheduler ()
#6  0x80563d9 in telnet ()
```

1. rappelons que ce serveur telnet de Heimdal avait bien fonctionné avec tous les autres clients (Heimdal, MIT, KTelnet)

```
#7  0x8051b11 in tn ()
#8  0x8052595 in main ()
#9  0x804e490 in _start ()
(gdb)
```

Il semblerait que cette fonction `krb5_rd_rep` serve à parcourir et décrypter le contenu d'un *buffer*, en écrivant le résultat dans un autre *buffer* qu'elle alloue elle-même dynamiquement. Peut-être le *segmentation fault* provient-il d'une erreur d'estimation de la taille d'allocation ? Il s'agirait dans ce cas d'un problème de type *buffer overflow*.

Annexe H

Patch pour les authentifications implicites

Ce fichier nous a été soumis par Douglas E. Engert (*deengert@anl.gov*). Son utilité et son comportement sont décrits dans la liste des tâches à accomplir, page 64.

```
*** ,an_to_ln.c Wed Jan  9 16:27:58 2002
--- an_to_ln.c  Fri Jan 11 15:44:09 2002
***************
*** 59,64 ****
--- 59,67 ----
  #define       KDBM_FETCH(db, key)     dbm_fetch(db, key)
  #endif /*ANAME_DB*/

+ krb5_error_code KRB5_CALLCONV
+ krb5_get_default_realm_equiv(krb5_context, char **, krb5_data *);
+
  /*
   * Find the portion of the flattened principal name that we use for mapping.
   */
***************
*** 606,614 ****

      realm_length = krb5_princ_realm(context, aname)->length;

!     if ((retval = krb5_get_default_realm(context, &def_realm))) {
        return(retval);
      }
      if (((size_t) realm_length != strlen(def_realm)) ||
          (memcmp(def_realm, krb5_princ_realm(context, aname)->data, realm_length))) {
          free(def_realm);
--- 609,622 ----

      realm_length = krb5_princ_realm(context, aname)->length;
```

```
!     if ((retval = krb5_get_default_realm_equiv(context, &def_realm,
!                                         krb5_princ_realm(context, aname)))) {
        return(retval);
      }
+       {
+
+
+       }
      if (((size_t) realm_length != strlen(def_realm)) ||
          (memcmp(def_realm, krb5_princ_realm(context, aname)->data, realm_length))) {
          free(def_realm);
***************
*** 673,679 ****
      /*
       * First get the default realm.
       */
!     if (!(kret = krb5_get_default_realm(context, &realm))) {
        /* Flatten the name */
        if (!(kret = krb5_unparse_name(context, aname, &pname))) {
            if ((mname = aname_full_to_mapping_name(pname))) {
--- 681,688 ----
      /*
       * First get the default realm.
       */
!     if (!(kret = krb5_get_default_realm_equiv(context, &realm,
!                                         krb5_princ_realm(context,aname)))) {
        /* Flatten the name */
        if (!(kret = krb5_unparse_name(context, aname, &pname))) {
            if ((mname = aname_full_to_mapping_name(pname))) {
***************
*** 803,805 ****
--- 812,869 ----
      return(kret);
  }

+ /*
+  Return either the default realm, or the requested realm
+  if the requested realm is considered an equiv_realm
+  to the default realm.
+  This is controled by the krb5.conf
+  [realms]->localrealm->equiv_realm
+  will allow relaxed access from the user's realm.
+ */
+ krb5_error_code KRB5_CALLCONV
+ krb5_get_default_realm_equiv(context, ret_realm, req_realm)
+       krb5_context context;
+       char **ret_realm;
+       krb5_data *req_realm;
+ {
+       krb5_error_code retval;
```

```
+       const char  *hierarchy[4];
+       char        **mapping_values = NULL;
+       int         i, nvalid;
+       char * def_realm;
+       int found = FALSE;
+
+       if (retval = krb5_get_default_realm(context, &def_realm)) {
+               return(retval);
+       }
+
+       hierarchy[0] = "realms";
+       hierarchy[1] = def_realm;
+       hierarchy[2] = "equiv_realm";
+       hierarchy[3] = (char *) NULL;
+       if (!(retval = profile_get_values(context->profile,
+                       hierarchy,
+                       &mapping_values))) {
+               /* We found one or more explicit mappings. */
+               for (i=0; mapping_values[i]; i++){
+                       if ((strlen(mapping_values[i]) == req_realm->length)
+                                       && !memcmp(mapping_values[i],
+
req_realm->data,req_realm->length)){
+                                found = TRUE;
+                                break;
+                       }
+               }
+       }
+       if (mapping_values)
+               profile_free_list(mapping_values);
+       if (found) {
+               free(def_realm);
+               if (!(def_realm = malloc(req_realm->length + 1))) {
+                       return (ENOMEM);
+               }
+               memcpy(def_realm, req_realm->data, req_realm->length);
+               def_realm[req_realm->length] = '\0';
+       }
+       *ret_realm = def_realm;
+       return 0;
+ }
```